Probabilities and Priorities in Timed CSP

by

Gavin Lowe

Technical Monograph PRG-111
ISBN 0-902928-88-0

November 1993

Oxford University Computing Laboratory
Programming Research Group
Wolfson Building
Parks Road
Oxford OX1 3QD
England

Oxford University Computing Laboratory
Programming Research Group
Wolfson Building
Parks Road
Oxford OX1 3QD
England

Electronic mail: gavin.lowe@comlab.ox.ac.uk

Probabilities and Priorities in Timed CSP

Gavin Lowe

St Hugh's College

A thesis submitted for the degree of Doctor of Philosophy at the University of Oxford, Hilary Term, 1993

Abstract

In this thesis we present two languages that are refinements of Reed and Roscoe's language of Timed CSP: a probabilistic language, and a prioritized language.

We begin by describing the prioritized language and its semantic model. The syntax is based upon that of Timed CSP except some of the operators are refined into biased operators. The semantics for our language represents a process as the set of its possible behaviours, where a behaviour models the priorities for different actions. A number of algebraic laws for our language are given and the model is illustrated with two examples.

We then describe the probabilistic language, which is built on top of the prioritized language. The only cause of nondeterminism in the prioritized language is the nondeterministic choice operator; by replacing this with a probabilistic choice operator we obtain a language where it is possible to calculate the probability of any particular behaviour. We produce a semantic model for our language, which gives the probabilities of different behaviours occurring, as well as modelling the relative priorities for events within a behaviour. The model is illustrated with an example of a communications protocol transmitting messages over an unreliable medium.

A complete compositional proof system is presented for the prioritized language, which can be used for proving behavioural specifications are met. This proof system can also be used to prove non-probabilistic specifications are met by probabilistic processes, via an abstraction theorem between the two models.

An abstraction theorem is presented relating the Prioritized Model to the Timed Failures Model. This enables unprioritized processes to be refined into prioritized ones.

Finally a compositional proof system is presented for the probabilistic language. This can be used to prove specifications such as "an a becomes available within two seconds with a probability of at least 90%". Unfortunately proofs of probabilistic specifications are considerably more difficult than in the unprobabilistic case. We examine these difficulties and show how they can be overcome. The proof system is illustrated with an example of a communications protocol transmitting over an unreliable medium: we examine the probability of a message being correctly transmitted within a given time.

Acknowledgements

I would like to thank my supervisor, Bill Roscoe, for his advice throughout the preparation of this thesis. He has provided many useful suggestions, and pointed out errors when I was going astray.

I must thank my examiners, Bengt Jonsson and He Jifeng, for making a number of suggestions as to how the presentation of this thesis could be improved. Karen Seidel made many useful comments on an earlier paper on this subject.

Brian Scott and Steve Schneider have provided useful sounding boards for new ideas. I have enjoyed many fruitful conversations with Jim Davies about the form of languages for specifying timed communicating processes. I have also benefited from discussions with Tony Hoare and Mike Reed. I have received many interesting comments and ideas from my colleagues on the SPEC and REACT projects.

Thanks must also go to my office colleagues, especially Nacho, Augusto, Brian, Mat, Janet, Bryan, Andrew, Katherine and Sharon, for making the attic a pleasant place to work, and for their friendship and support.

This work was supported by a grant from the Science and Engineering Research Council of the United Kingdom and a scholarship from St. Hugh's College, Oxford.

Finally I must thank my friends in Oxford University Cave Club for regularly dragging me underground, and providing much needed breaks from my work.

Contents

Chapter 1

Introduction

Communicating Sequential Processes [Hoa85] is a language for reasoning about concurrent processes. This model has been extended [RR86, RR87, Ree88] to include a treatment of timing information. Previous models have allowed nondeterminism; this has proved to be a useful tool in that it allows one to underspecify the behaviour of processes, and so maintain a high level of abstraction. However, previous models have failed to model the probabilities involved in nondeterministic choices. In this thesis we aim to overcome this deficiency, and in doing so also produce a model with a notion of priority.

We believe that it is important to be able to model probabilistic behaviour for a number of reasons.

- Many components of computer systems display behaviour that is probabilistic in nature. For example, communication media can often corrupt or lose messages; it is reasonable to model such a medium as a process that acts unreliably with a certain probability. Suppose we have a communications protocol that transmits messages over such a medium. We would like to be able to prove results such as "the message is correctly transmitted within 3 seconds with a probability of 99%". In order to do this we need to be able to model the probabilities of messages being lost or corrupted by the medium.

- There are many problems in computer science that cannot be solved efficiently by a deterministic algorithm but for which there exist efficient probabilistic algorithms. Examples include consensus protocols [AH90, Sei92], mutual exclusion [PZ86], and self stabilization [Her90].

- We often want to consider a process operating in an environment that behaves in a manner that could be considered probabilistic. For example, consider a server providing a service to several clients, where each client may request the use of the server and then release it when it has finished. Here the clients can be considered as forming the environment of the server. If we abstract away from the details of the behaviours of the clients then it is reasonable to model them as agents that make requests for service with a frequency governed by some probability distribution. We need to be able to model these probabilities in order to prove results relating, say, to the probability of the server reacting to a request for service within a given amount of time.

We believe that a prioritized model is a useful thing because this will give us a more powerful language for specifying processes. Certain applications naturally require different actions to have different priorities:

- When we model an interrupt mechanism we would like the interrupt event to have a higher priority than what it is interrupting; otherwise the interrupt event could be ignored. This is illustrated in section 3.2.2.
- Priorities are useful when modelling an arbitration protocol for dealing with the case where several clients compete for the use of a resource. Arbitration can be achieved by giving different priorities to the different clients. If it is desired to have a fixed hierarchy — for example if the clients can be ranked in order of importance — then these priorities can be constant through time. Alternatively the priorities can be varied so as to achieve fairness: we illustrate this in section 3.2.1 where we model a lift system which gives different priorities to requests from different floors in such a way that the lift is guaranteed to arrive at a floor within a certain time of being requested.

A prioritized language is also useful because if we remove the nondeterministic choice operator we are left with a completely deterministic language. Nondeterminism can be considered a bad thing, in that a nondeterministic process is unpredictable and we would like programs that we write to always behave in a predictable way: this will be true of any program written in our deterministic language.

The probabilistic language is produced from the prioritized language by replacing the nondeterministic choice operator by a probabilistic choice operator. Thus, the only cause of nondeterminism in the probabilistic language is the probabilistic choice operator. We have chosen to build our probabilistic model upon this prioritized model because it is our belief that in order to argue about probabilistic behaviours it is necessary to be able to predict precisely how the non-probabilistic parts behave in a given circumstance. If a language includes other forms of nondeterminism, besides probabilistic choice, then it is not possible to predict the probability of a particular behaviour occurring. For example, consider the question:

> What is the probability that the process $a \longrightarrow STOP \quad b \longrightarrow STOP$ performs an a if the environment is willing to perform either an a or a b at time 0?

In the standard models of Timed CSP the external choice operator is underspecified, and so it is not possible to answer this question. We need to refine this operator in order to produce a deterministic version. In particular we will define two new external choice operators: a left-biased choice operator written $\Box$ and a right-biased choice operator written $\Box$. These will respectively arbitrate in favour of their left- or right-hand arguments when the environment is willing to perform actions from either side; in the circumstances described above the process $a \longrightarrow STOP \Box b \longrightarrow STOP$ will perform an a, whereas $a \longrightarrow STOP \Box b \longrightarrow STOP$ will perform a b.

Some workers have got around the problem of the underspecification of the external choice operator by insisting that if a process is able to perform two or more separate actions then the choice is made by the environment. We avoid this because:

- we consider the environment to be a more passive entity than the process: it seems strange that an environment is able to choose between two actions whereas a process is not;

- this idea clashes with our intuition of a system (built out of smaller components) being in an environment consisting of a user who is willing to observe any event.

Most previous probabilistic process algebras have used a probabilistic external choice operator, written say as $_p \; _q$, such that $P \, _p \; _q \, Q$ offers the environment a choice between the actions of P and Q; if the environment is willing to perform the actions of either, then P is chosen with probability p and Q is chosen with probability q (where $p + q = 1$). We choose to separate the two phenomena of external choice and probabilistic nondeterminism for we believe them to be orthogonal issues. Our language will include two deterministic (prioritized) external choice operators and a probabilistic internal choice operator. Having more operators produces a language that, while being harder to reason about, is easier to reason with. Our prioritized external choice operators will be the same as the operators $_1 \; _0$ and $_0 \; _1$: hence in a sense the prioritized external choice is the "limit" of a probabilistic external choice. The probabilistic external choice operator can be regained from the prioritized operators via the identity $P \, _p \; _q \, Q = (P \boxdot Q) \, _p\sqcap_q \, (P \boxdot Q)$, where $_p\sqcap_q$ is a probabilistic internal choice operator.

The rest of this thesis is structured as follows. In chapter 2 we give a brief review of the Timed Failures Model of Timed CSP. In chapter 3 we describe our prioritized language and its semantic model. We will represent a process by the set of behaviours that it can perform. We will represent a behaviour, or observation, by a triple $(\tau, \sqsubseteq, s)$ where τ is the time that the observation ends, $\sqsubseteq$ records the different priorities given to actions during the behaviour, and s records the events performed. We give semantic definitions for all the constructs of the language and prove the definitions sound with respect to a number of healthiness conditions for the semantic space. In section 3.6 we show how, by removing the nondeterministic choice operator from the syntax, we can produce a completely deterministic language.

In chapter 4 we consider the probabilistic language. The syntax is the same as the syntax for the prioritized language except the nondeterministic choice operator is replaced by a probabilistic choice operator: the process $P \, _p\sqcap_q \, Q$ acts like P with probability p and like Q with probability q. The semantic model represents a process by a pair (A, f) where A is the set of behaviours that it can perform and f is a function that gives the probability of each behaviour occurring given a suitable environment.

In chapter 5 we examine ways of proving properties of prioritized processes. We write P **sat** $S(\tau, \sqsubseteq, s)$, where $S(\tau, \sqsubseteq, s)$ is a specification whose argument represents a behaviour, to specify that *all* behaviours of the process P satisfy S. We then describe a specification language based upon the one in [Dav91]. The syntax of the specification language is as near as possible to the English language so that we can be reasonably confident that our specifications meet our informal requirements. We present a complete compositional proof system, in the style of [DS89b], consisting of a number of inference rules. For composite processes, a proof obligation is broken down into proof obligations on the subcomponents. We illustrate the proof system with an example. This proof system can also be used to prove properties of probabilistic processes: we can prove that all behaviours of a probabilistic process satisfy a specification by showing that all behaviours of the corresponding unprobabilistic process satisfy the same specification.

In chapter 6 we relate the Prioritized Model to the Timed Failures Model. We investigate which failures could have resulted from a particular prioritized behaviour, and thus produce an abstraction result from the Prioritized Model to the Failures Model. We then show how this result can be use to prove properties of prioritized processes. We will show that if a process in the Timed Failures Model satisfies a specification then all its prioritized refinements satisfy a related specification.

In chapter 7 we give a proof system for proving properties of probabilistic processes. We write $P \ \mathbf{sat}_{\rho}^{\geqslant p} \ S(\tau, \sqsubseteq, s)$ to specify that, whatever the environment offers, the probability that process P performs a behaviour $(\tau, \sqsubseteq, s)$ that satisfies the predicate $S(\tau, \sqsubseteq, s)$ is at least p. We will also define conditional probabilities: we will write $P \ \mathbf{sat}_{\rho}^{\geqslant p} \ S(\tau, \sqsubseteq, s) \mid G(\tau, \sqsubseteq, s)$ to specify that the probability that P performs a behaviour that satisfies S *given that* it satisfies G is at least p. Unfortunately, proving properties of probabilistic processes is considerably harder than for unprobabilistic processes; we explain what the main difficulties are and how these can be overcome. We illustrate the proof system via a case study of a protocol transmitting messages over an unreliable medium. We show that the protocol acts like a buffer, and perform an analysis of its performance: we prove a result that gives the probability of a message being correctly transmitted within a certain amount of time.

In order to keep this thesis to a reasonable size we have omitted a number of proofs that have appeared elsewhere. The interested reader is referred to the relevant papers.

Chapter 2

Timed CSP

In this chapter we give a brief overview of the syntax and semantics of Timed CSP. The first models appeared in [RR86, RR87] and [Ree88]. These have since been extended in [Sch90], [Dav91] and [DS92a]. The forthcoming book on CSP [DRRS93] will provide a complete overview. The model described here fits most closely with the model described in [Dav91], although the specification language we present is nearer to that of [DRRS93].

The development of a mathematical model of a CSP-based language normally follows a particular approach:

- A mathematical structure is described for representing a particular behaviour or observation of a process; a process is then represented by the set of such behaviours that it can perform.

- Semantic definitions are given for all the constructs of the language: these define precisely what behaviours a process can perform; for composite processes, the semantic definition is in terms of the semantic representations of the subcomponents.

- Certain healthiness conditions are conjectured: these express properties that we would expect all processes to have, and outlaw pathological processes. Proving that our definitions meet these healthiness conditions improves our confidence in the model. Alternatively, if we find that a healthiness condition is not satisfied by one of our semantic definitions then we know something is wrong — perhaps because the definition is wrong, or perhaps because the mathematical structure we are using to represent processes does not carry enough information.

  The healthiness conditions are often used in proving results about specific processes, and in proving algebraic laws — they give us extra information about how processes behave.

- A proof system, consisting of a number of proof rules, is developed for the language: rules are given for proving properties of atomic processes directly; for composite processes, a proof rule is given that reduces a proof obligation to proof obligations on the subcomponents. These rules are proved sound with respect to the semantic definitions.

- The semantic model is related to simpler semantic models: this improves our confidence that our model "agrees" with existing models, and also provides a useful proof technique — properties can be proved to hold of processes by arguing in the simpler model.

This is the approach we will take in developing the models in this thesis.

The semantic models of Timed CSP are based on a number of assumptions which we list here; the Prioritized and Probabilistic Models presented in this thesis will be based upon much the same assumptions.

Communication Communication between processes is achieved via handshaking: observable events can only be performed with the cooperation of the environment.

Real time We model time using the non-negative real numbers. There is no lower bound between the times of consecutive independent events. Each observation is made with respect to a global clock: this clock cannot be accessed by any process.

Instantaneous events Events have zero duration; if we want to model an action with a significant duration then we should model the start and finish as two distinct events.

Non-Zenoness We assume that no process may make an infinite amount of progress in a finite time.

Maximum progress If a process and its environment are both willing to perform an event, then the process may not idle: it must either perform this event or some other event (visible or invisible). In the Prioritized Model presented in chapter 3 we will make the assumption that a process performs the action offered by the environment to which it gives highest priority.

Hidden events When events are hidden they do not require the cooperation of the environment and so occur as soon as the process is ready for them. In the Prioritized Model we will make the assumption that the process performs the number of internal events that it gives highest priority to. This and the previous assumption can be considered as maximal progress assumptions: the process performs as many events (internal or external) as the environment allows; in the Prioritized Model the process performs whichever action it gives highest priority to.

Causality There is a non-zero delay between consecutive events in sequential processes, so immediate causality is not allowed. The reader should note that the most recent models of Timed CSP *do* allow immediate causality. For simplicity we do not allow immediate causality in this thesis.

2.1 Syntax of Timed CSP

The syntax of Timed CSP is as follows:

$$\begin{array}{lll}
P ::= & STOP \mid SKIP \mid WAIT\ t \mid X \mid & \text{basic processes} \\
& a \xrightarrow{t} P \mid P \quad P \mid WAIT\ t\, ; P \mid & \text{sequential composition} \\
& P \sqcap P \mid \quad {}_{i \in I} P_i \mid P \quad P \mid c?d : D \xrightarrow{t_d} P_d \mid & \text{alternation} \\
& P \parallel P \mid P\ {}^{A}\!\|^{B}\ P \mid P \quad P \mid P \underset{A}{\|} P \mid & \text{parallel composition} \\
& P \setminus A \mid f(P) \mid f^{-1}(P) \mid & \text{abstraction and renaming} \\
& P \ ^{t}\ P \mid P \ _{t}\ P \mid P \underset{a}{\nabla} P \mid & \text{transfer operators} \\
& \mu X \quad P \mid \mu X \quad P \mid \langle X_i = P_i \rangle_j & \text{recursion}
\end{array}$$

where t and t_a range over the set $TIME$ of times, which we take to be non-negative real numbers; a ranges over some alphabet Σ of events; and A and B range over $\quad \Sigma$. X ranges over process names. c ranges over the set $CHAN$ of channels, D ranges over datatypes, and d ranges over D. I is an index set ranged over by i and j. The renaming function f ranges over functions of type $\Sigma \to \Sigma$.

$STOP$ represents the deadlocked process that can perform no visible events. The process $SKIP$ can do nothing except terminate by performing the event . $WAIT\ t$ can terminate after t time units. The variable X represents a call to the process bound to X.

The process $a \xrightarrow{t} P$ is initially willing to perform the visible event a; once it has performed an a, it will act like process P after a delay of length t. If we omit the parameter t we will take its value to be δ — a system constant. The process $P \quad Q$ will initially act like P; if P terminates, the process will then act like Q after a delay of length δ. $WAIT\ t\, ; P$ acts like P, delayed by t time units.

$P \sqcap Q$ nondeterministically chooses between the processes P and Q. The process $\quad {}_{i \in I} P_i$ nondeterministically chooses between the processes P_i indexed by the set I. The process $P \quad Q$ offers the environment a choice between the two processes P and Q: as soon as the environment is willing to perform an event offered by one of the processes, that process is chosen. If the environment is first able to perform an event offered by P at the same time as it is first able to perform an event offered by Q, then the choice is made nondeterministically. Communication of values along channels is modelled by the process $c?d : D \xrightarrow{t_d} P_d$: this is initially willing to input any value d of type D on channel c, and then after a delay of length t_d act like process P_d.

The process $P \parallel Q$ executes P and Q in lockstep parallel, synchronizing on every visible event. The process $P\ {}^{A}\!\|^{B}\ Q$ executes P and Q in parallel; P can only perform events from the set A, and Q can only perform events from the set B; they must synchronize on events from the set $A \cap B$. This processes is normally written as $P\ {}_{A}\|_{B}\ Q$ with the alphabets as subscripts; in this thesis we write alphabets as superscripts because we want to use subscripts for probabilities. $P \quad Q$ interleaves P and Q: the two processes are executed in parallel without synchronization; if the environment is able to do events of P or of Q, but not both, then the choice is made nondeterministically. The process $P \underset{C}{\|} Q$ is a hybrid parallel operator: it forces synchronisation on the events from C, but allows interleaving on all other events.

Abstraction is achieved via the hiding operator: the process $P \setminus A$ acts like P except all the events from the set A occur silently: the environment's cooperation is not necessary for the events from A to occur, so they happen as soon as the process is able to perform them. The process $f(P)$ acts like P except all the external events are renamed by the function f. The process $f^{-1}(P)$ acts like P except it performs the event a whenever P can perform $f(a)$.

Timeouts are modelled using the operator: $P \;^{t}\; Q$ initially acts like P; if no visible event has been performed by time t, then it times out, and after a delay of length δ acts like Q. With the process $P \;_{t}\; Q$, control is transferred from P to Q at time t, with a delay of δ, regardless of the progress P has made up until this time. Interrupts are modelled using the ∇ operator: $P \underset{a}{\nabla} Q$ initially acts like P except at any time it is willing to perform the interrupt event a: if an a is performed, control is transferred to the interrupt handler Q.

The processes $\mu X \;\; P$ and $\mu X \;\; P$ are recursive processes. They both act like P, with X representing a recursive call. With $\mu X \;\; P$, there is a delay of length δ associated with all recursive calls; with $\mu X \;\; P$, the recursion is immediate: it is the responsibility of the programmer to ensure that the process cannot perform infinitely many recursions in a finite time. Mutual recursion is modelled by $\langle X_i = P_i \rangle_j$; this represents the jth component of the vector of processes $\langle X_i \mid i \in I \rangle$ mutually defined by the set of equations $\{X_i = P_i \mid i \in I\}$.

2.2 The Timed Failures Model

2.2.1 Timed failures

A timed event is a pair (t, a) where t is a member of the set $TIME$ of times, which we take to be non-negative real numbers, and a is a member of the set Σ of visible actions.

A timed trace is a finite sequence of timed events arranged in non-decreasing order of times. For example the trace $\langle (1, a), (2, b), (2, c) \rangle$ represents the performance of an a at time 1, and a b and a c at time 2. We write $T\Sigma$ for the set of timed events, $T\Sigma^*_\leq$ for the set of timed traces, and s for a typical member of $T\Sigma^*_\leq$:

$$T\Sigma \mathrel{\widehat{=}} TIME \times \Sigma \qquad T\Sigma^*_\leq \mathrel{\widehat{=}} \{s \in \mathrm{seq}(T\Sigma) \mid (t, a) \text{ precedes } (t', b) \text{ in } s \Rightarrow t \quad t'\}$$

If a process is unwilling to perform a particular timed event then we say that it can be refused. A refusal $\aleph$ is a set of events that are seen to be refused by a process. Our assumptions about finite speed of processes allow us to restrict our attention to sets of refusals that are the union of a finite number of refusal tokens:

$$RSET \mathrel{\widehat{=}} \{\bigcup C \mid C \in \;\; (RTOK)\}$$

where a refusal token is the cross product of a half open time interval and a set of events:

$$RTOK \mathrel{\widehat{=}} \{I \times A \mid I \in HOTINT \land A \in \;\; \Sigma\} \qquad HOTINT \mathrel{\widehat{=}} \{[t, t') \mid t, t' \in TIME \land t < t'\}$$

A timed failure is a pair $(s, \aleph)$ where s is a timed trace and $\aleph$ is a refusal set:

$$TF \mathrel{\widehat{=}} T\Sigma^*_\leq \times RSET$$

The pair $(s, \aleph)$ represents that the process performs the events in s while refusing to perform the events in $\aleph$. For example, the timed failure $(\langle(1, a), (2, b)\rangle, [2, 3) \times \{b, c\})$ represents an observation where an a occurs at time 1 and a b at time 2, and the process refuses a b and a c during the interval $[2, 3)$. Note that a timed event can appear in both the trace and the refusal: in the example the process performs one b at time 2 but refuses to perform any more bs.

2.2.2 Notation

In this section we describe the notation we will use for reasoning about timed failures. An index of notation appears on pages 214–218.

We use the following notation for traces: the empty trace is denoted by $\langle\rangle$; concatenation of traces is written using ; we write s_1 in s_2 if s_1 is a contiguous subsequence of s_2; we write $s_1 \cong s_2$ if s_1 is a permutation of s_2.

The function *times* returns the set of all times at which events are performed or refused:

$$times\, s \mathrel{\widehat=} \{t \mid \exists a \quad \langle(t, a)\rangle \text{ in } s\} \qquad times\, \aleph \mathrel{\widehat=} \{t \mid \exists a \quad (t, a) \in \aleph\}$$

We can use this to define *begin* and *end* functions that return the time of the beginning or end of a trace:

$$\begin{aligned} begin\langle\rangle &\mathrel{\widehat=} \infty & \qquad begin\, s &\mathrel{\widehat=} \inf(times\, s) \quad \text{if } s \neq \langle\rangle \\ end\langle\rangle &\mathrel{\widehat=} 0 & end\, s &\mathrel{\widehat=} \sup(times\, s) \quad \text{if } s \neq \langle\rangle \end{aligned}$$

Similar functions can be defined for refusals and observations:

$$\begin{aligned} begin\{\} &\mathrel{\widehat=} \infty & \qquad begin\, \aleph &\mathrel{\widehat=} \inf(times\, \aleph) \quad \text{if } \aleph \neq \{\} \\ end\{\} &\mathrel{\widehat=} 0 & end\, \aleph &\mathrel{\widehat=} \sup(times\, \aleph) \quad \text{if } \aleph \neq \{\} \\ begin(s, \aleph) &\mathrel{\widehat=} \min\{begin\, s, begin\, \aleph\} & end(s, \aleph) &\mathrel{\widehat=} \max\{end\, s, end\, \aleph\} \end{aligned}$$

The values for the empty trace and empty refusal are chosen so as to make the subsequent mathematics as simple as possible.

The functions *first* and *last* return the first and last events from a trace; for the empty trace they return the non-event ε:

$$\begin{aligned} first\langle\rangle &\mathrel{\widehat=} \varepsilon & \qquad first(\langle(t, a)\rangle\ \ s) &\mathrel{\widehat=} a \\ last\langle\rangle &\mathrel{\widehat=} \varepsilon & last(s\ \ \langle(t, a)\rangle) &\mathrel{\widehat=} a \end{aligned}$$

The functions *head* and *foot* return the first and last timed events from a trace:

$$head\, s \mathrel{\widehat=} (begin\, s, first\, s) \qquad foot\, s \mathrel{\widehat=} (end\, s, last\, s)$$

The *during* operator ($\uparrow$) returns the part of a trace or refusal occurring during some time interval I:

$$\begin{aligned} \langle\rangle \uparrow I &\mathrel{\widehat=} \langle\rangle \\ (\langle(t, a)\rangle\ \ s) \uparrow I &\mathrel{\widehat=} \begin{cases} \langle(t, a)\rangle\ \ (s \uparrow I) & \text{if } t \in I \\ s \uparrow I & \text{if } t \notin I \end{cases} \\ \aleph \uparrow I &\mathrel{\widehat=} \{(t, a) \in \aleph \mid t \in I\} \end{aligned}$$

Note that in order to make $\aleph \uparrow I$ a member of $RSET$ we will normally take I to be a finite union of half open intervals. We can use the during operator to define *before* ( ), *strictly before* ( ), *after* ( ), *strictly after* ( ), and *at* ($\uparrow$) operators:

$$
\begin{array}{rcl@{\qquad}rcl}
s \quad t & \mathrel{\widehat=} & s \uparrow [0,t] & \aleph \quad t & \mathrel{\widehat=} & \aleph \uparrow [0,t] \\
s \quad t & \mathrel{\widehat=} & s \uparrow [0,t) & \aleph \quad t & \mathrel{\widehat=} & \aleph \uparrow [0,t) \\
s \quad t & \mathrel{\widehat=} & s \uparrow [t,\infty) & \aleph \quad t & \mathrel{\widehat=} & \aleph \uparrow [t,\infty) \\
s \quad t & \mathrel{\widehat=} & s \uparrow (t,\infty) & \aleph \quad t & \mathrel{\widehat=} & \aleph \uparrow (t,\infty) \\
s \uparrow t & \mathrel{\widehat=} & s \uparrow \{t\} & \aleph \uparrow t & \mathrel{\widehat=} & \aleph \uparrow \{t\}
\end{array}
$$

The *restrict* operator ( ) restricts a trace or a refusal to events from a particular set:

$$
\begin{array}{rcl}
\langle\rangle \quad A & \mathrel{\widehat=} & \langle\rangle \\
(\langle(t,a)\rangle \quad s) \quad A & \mathrel{\widehat=} & \begin{cases} \langle(t,a)\rangle \quad (s \quad A) & \text{if } a \in A \\ s \quad A & \text{if } a \notin A \end{cases} \\
\aleph \quad A & \mathrel{\widehat=} & \{(t,a) \in \aleph \mid a \in A\}
\end{array}
$$

The *hiding* operator ($\backslash$) restricts a trace or a refusal to all events *not* in a certain set:

$$
s \setminus A \mathrel{\widehat=} s \quad (\Sigma \setminus A) \qquad \aleph \setminus A \mathrel{\widehat=} \aleph \quad (\Sigma \setminus A)
$$

Traces and refusals can be relabelled by a function $f : \Sigma \to \Sigma$ in the obvious way:

$$
\begin{array}{rcl}
f(\langle\rangle) & \mathrel{\widehat=} & \langle\rangle \\
f(\langle(t,a)\rangle \quad s) & \mathrel{\widehat=} & \langle(t,f(a))\rangle \quad f(s) \\
f(\aleph) & \mathrel{\widehat=} & \{(t,f(a)) \mid (t,a) \in \aleph\} \\
f^{-1}(\aleph) & \mathrel{\widehat=} & \{(t,a) \mid (t,f(a)) \in \aleph\}
\end{array}
$$

The alphabet operator (Σ) returns the set of untimed events from a trace or refusal:

$$
\Sigma s \mathrel{\widehat=} \{a \mid \exists t \quad \langle(t,a)\rangle \textbf{ in } s\} \qquad \Sigma\aleph \mathrel{\widehat=} \{a \mid \exists t \quad (t,a) \in \aleph\}
$$

The operators $+$ and $-$ are used to temporally shift traces, forwards or backwards through time:

$$
\begin{array}{rcl}
\langle\rangle + t & \mathrel{\widehat=} & \langle\rangle \\
(\langle(t',a)\rangle \quad s) + t & \mathrel{\widehat=} & \langle(t'+t,a)\rangle \quad (s+t) \\
\langle\rangle - t & \mathrel{\widehat=} & \langle\rangle \\
(\langle(t',a)\rangle \quad s) - t & \mathrel{\widehat=} & \begin{cases} \langle(t'-t,a)\rangle \quad (s-t) & \text{if } t' \quad t \\ s - t & \text{if } t' < t \end{cases}
\end{array}
$$

These operators can also be applied to refusals or behaviours:

$$
\begin{array}{rcl}
\aleph + t & \mathrel{\widehat=} & \{(t'+t,a) \mid (t',a) \in \aleph\} \\
\aleph - t & \mathrel{\widehat=} & \{(t'-t,a) \mid (t',a) \in \aleph \wedge t' \quad t\} \\
(s,\aleph) + t & \mathrel{\widehat=} & (s+t, \aleph+t) \\
(s,\aleph) - t & \mathrel{\widehat=} & (s-t, \aleph-t)
\end{array}
$$

2.2.3 The Timed Failures Model

The Timed Failures Model represents a Timed CSP process by the set of behaviours that it can perform. We define S_{TF} to be the set of all timed failures:

$$S_{TF} \mathrel{\hat{=}} \quad (TF)$$

The Timed Failures Model $\mathcal{M}_{TF}$ is then defined to be those members S of S_{TF} satisfying the following seven healthiness conditions:

1. $(\langle\rangle, \{\}) \in S$
2. $(s \quad w, \aleph) \in S \Rightarrow (s, \aleph \quad \mathit{begin}\, w) \in S$
3. $(s, \aleph) \in S \wedge s \cong w \Rightarrow (w, \aleph) \in S$
4. $(s, \aleph) \in S \wedge t \quad 0 \Rightarrow$
   $\exists \aleph' \in RSET \quad \aleph \subseteq \aleph' \wedge (s, \aleph') \in S$
   $\wedge (\forall t' \quad t' \quad t \wedge (t', a) \notin \aleph' \Rightarrow (s \quad t' \quad (t', a), \aleph' \quad t') \in S)$
5. $\forall t \in [0, \infty) \quad \exists\, n(t) \in \quad \forall (s, \aleph) \in S \quad \mathit{end}\, s \quad t \Rightarrow \#s \quad n(t)$
6. $(s, \aleph) \in S \wedge \aleph' \in RSET \wedge \aleph' \subseteq \aleph \Rightarrow (s, \aleph') \in S$
7. $\begin{pmatrix} (s \quad w, \aleph) \in S \wedge \aleph' \in RSET \\ \wedge\, \mathit{end}\, s \quad \mathit{begin}\, \aleph' \wedge \mathit{end}\, \aleph' \quad \mathit{begin}\, w \\ \wedge\, \forall (t, a) \in \aleph' \quad (s \quad (t, a), \aleph \quad t) \notin S \end{pmatrix} \Rightarrow (s \quad w, \aleph \cup \aleph') \in S$

The first condition says that every process can perform the empty trace and refuse nothing. The second condition says that if any particular behaviour can be observed, then any prefix of that behaviour can also be observed. The third condition states that simultaneous events in a trace can be reordered.

Condition 4 says that any refusal set $\aleph$ can be enlarged to a maximal refusal set $\aleph'$ that contains *all* timed events that the process cannot perform during this behaviour. The fact that $\aleph'$ is a member of the set $RSET$ of refusals relates to our finite speed assumption: the set of events that the process cannot perform changes only finitely often in finite time. The fifth condition also relates to our finite speed assumption: there is a bound $n(t)$ on the number of events that the process can perform within time t.

Condition 6 says that if the process can refuse all the events of $\aleph$ then it can also refuse any subset of $\aleph$. The final condition says that if the refusal set $\aleph'$ is such that all of its elements occur between the times of the traces s and w, and none of the events can be performed after trace s, then $\aleph'$ can be added to the refusal set.

We place a metric upon the set of timed failures by considering the first time at which two elements can be distinguished. For $S \in S_{TF}$ we define

$$S \quad t \mathrel{\hat{=}} \{(s, \aleph) \in S \mid \mathit{end}(s, \aleph) \quad t\}$$

We then define the metric d by

$$d(S, T) \mathrel{\hat{=}} \inf(\{2^{-t} \mid S \quad t = T \quad t\} \cup \{1\})$$

This metric will be used to give a semantics to recursive processes.

2.2.4 The semantic function

The syntax of Timed CSP includes the term X: a variable that can be bound to a process. In order to give a semantics to variables, we define a space ENV_F of environments or variable bindings:

$$ENV_F \mathrel{\hat{=}} VAR \rightarrow S_{TF}$$

We will write $\rho\ X$ for the value assigned to variable X in environment ρ.

We can now define the semantic function:

$$\mathcal{F}_T : TCSP \rightarrow ENV_F \rightarrow S_{TF}$$

$\mathcal{F}_T\ P\ \rho$ will represent the set of timed failures that Timed CSP term P can perform, given variable binding ρ. If P is a process (i.e. if it has no free variables) then it makes sense to omit reference to the environment and simply to write $\mathcal{F}_T\ P$. In the next section we give semantic definitions for all the Timed CSP constructs.

2.3 Semantic definitions

2.3.1 Basic processes

The process $STOP$ can only perform the empty trace; it can refuse anything:

$$\mathcal{F}_T\ STOP\ \rho \mathrel{\hat{=}} \{(\langle\rangle, \aleph) \mid \aleph \in RSET\}$$

The process $WAIT\ t$ can perform the empty trace as long as it does not refuse a after time t; alternatively it can perform a at any time t' after t as long as it does not refuse a between t and t'.

$$\begin{aligned}\mathcal{F}_T\ WAIT\ t\ \rho \mathrel{\hat{=}} & \{(\langle\rangle, \aleph) \mid \quad \notin \Sigma(\aleph \quad t)\} \\ & \cup\{(\langle(t', \ )\rangle, \aleph) \mid t' \quad t \wedge \quad \notin \Sigma(\aleph \uparrow [t, t'))\}\end{aligned}$$

$SKIP$ is the same as $WAIT\ 0$ so we have the following definition:

$$\begin{aligned}\mathcal{F}_T\ SKIP\ \rho \mathrel{\hat{=}} & \{(\langle\rangle, \aleph) \mid \quad \notin \Sigma\aleph\} \\ & \cup\{(\langle(t, \ )\rangle, \aleph) \mid \quad \notin \Sigma(\aleph \quad t)\}\end{aligned}$$

The term X represents the process bound by the environment to the variable X:

$$\mathcal{F}_T\ X\ \rho \mathrel{\hat{=}} \rho\ X$$

2.3.2 Prefixing

The process $a \xrightarrow{t} P$ can perform the empty trace as long as it does not refuse an a; alternatively, it can perform an a at some time t', and then act like P starting from time $t'+t$, as long as it does not refuse an a before t'.

$$\begin{aligned}\mathcal{F}_T\ a \xrightarrow{t} P\ \rho \mathrel{\hat{=}} & \{(\langle\rangle, \aleph) \mid a \notin \Sigma\aleph\} \\ & \cup\{(\langle(t', a)\rangle \quad s_P + t' + t, \aleph) \mid a \notin \Sigma(\aleph \quad t') \wedge (s_P, \aleph - t' - t) \in \mathcal{F}_T\ P\ \rho\}\end{aligned}$$

2.3.3 Sequential composition

We assume that in the combination $P\ \ Q$ the event   is always available for P: in other words, P may terminate as soon as it is able. Hence $P\ \ Q$ can perform a non-terminating trace of P only if P is unwilling to perform a , i.e. if it is always able to refuse a . Similarly, P can terminate at time t only if it could refuse a   at all earlier times; in this case control is passed to Q starting from time $t+\delta$.

$$\begin{aligned}\mathcal{F}_T\ P\ \ Q\ \rho \mathrel{\hat=} & \{(s,\aleph) \mid\ \ \notin \Sigma s \land \forall I \in HOTINT\ \ (s, \aleph \cup (I \times \{\ \})) \in \mathcal{F}_T\ P\ \rho\} \\ & \cup\{(s,\aleph) \mid \exists t\ \ \ \notin \Sigma(s\ \ t) \land (s\ \ t\ \ (t,\ \ ), \aleph\ \ t \cup ([0,t) \times \{\ \})) \in \mathcal{F}_T\ P\ \rho \\ & \qquad \land s \uparrow (t, t+\delta) = \langle\rangle \land (s - t - \delta, \aleph - t - \delta) \in \mathcal{F}_T\ Q\ \rho\}\end{aligned}$$

The process $WAIT\ t\ ; P$ acts like P after a delay of length t:

$$\mathcal{F}_T\ WAIT\ t\ ; P\ \rho \mathrel{\hat=} \{(s+t,\aleph) \mid (s, \aleph - t) \in \mathcal{F}_T\ P\ \rho\}$$

2.3.4 Nondeterministic choice

$P \sqcap Q$ can act like either P or Q; similarly, $\ _{i \in I} P_i$ can act like any one of the P_i:

$$\begin{aligned}\mathcal{F}_T\ P \sqcap Q\ \rho &\mathrel{\hat=} \mathcal{F}_T\ P\ \rho \cup \mathcal{F}_T\ Q\ \rho \\ \mathcal{F}_T\ \ _{i \in I} P_i\ \rho &\mathrel{\hat=} \bigcup\{\mathcal{F}_T\ P_i\ \rho \mid i \in I\}\end{aligned}$$

This latter definition is sound only if the set of processes $\{P_i \mid i \in I\}$ is uniformly bounded in the following sense:

Definition 2.3.1: The set of processes $\{P_i \mid i \in I\}$ is *uniformly bounded* iff

$$\forall t : TIME\ ; \rho : ENV_P\ \ \exists n(t) :\ \ \forall i : I\ \ (s,\aleph) \in \mathcal{F}_T\ P_i\ \rho \land end\, s\ \ \ t \Rightarrow \#s\ \ \ n(t)$$

◇

The set is uniformly bounded if there is a uniform bound on the number of events that each process can perform within time t. This condition is necessary to ensure that condition 5 on the semantic space is satisfied.

2.3.5 External choice

The process $P\ \ Q$ offers the environment a choice between the events offered by P and Q. It can perform the empty trace when both P and Q can: in this case, every event refused by $P\ \ Q$ must be able to be refused by both P and Q. Similarly, if $P\ \ Q$ performs a nonempty trace of either P of Q then any event refused before the first visible event must be able to be refused by both P and Q.

$$\begin{aligned}&\mathcal{F}_T\ P\ \ Q \mathrel{\hat=} \\ &\quad \{(\langle\rangle,\aleph) \mid (\langle\rangle,\aleph) \in \mathcal{F}_T\ P\ \rho \cap \mathcal{F}_T\ Q\ \rho\} \\ &\quad \cup\{(s,\aleph) \mid s \neq \langle\rangle \land (s,\aleph) \in \mathcal{F}_T\ P\ \rho \cup \mathcal{F}_T\ Q\ \rho \land (\langle\rangle, \aleph\ \ begin\, s) \in \mathcal{F}_T\ P\ \rho \cap \mathcal{F}_T\ Q\ \rho\}\end{aligned}$$

The process $c?d : D \xrightarrow{t_d} P_d$ is able to input any value d of type D on channel c and then, after a delay of length t_d, act like P_d. If it performs the empty trace then it must not refuse to input from c. Alternatively it can input some value d at time t, and then act like P_d after a delay of length t_d, as long as it does not refuse to input from c before t.

$$\begin{aligned}
&\mathcal{F}_T\ c?d : D \xrightarrow{t_d} P_d\ \rho \mathrel{\widehat=} \\
&\quad \{(\langle\rangle, \aleph) \mid c.D \cap \Sigma\aleph = \{\}\} \\
&\quad \cup \{(\langle(t, c?d)\rangle\quad s + t + t_d, \aleph) \mid d \in D \wedge c.D \cap \Sigma(\aleph\quad t) = \{\} \wedge (s, \aleph - t - t_d) \in \mathcal{F}_T\ P_d\ \rho\}
\end{aligned}$$

This definition is sound if the set of processes $\{P_d \mid d \in D\}$ is uniformly bounded in the sense of the previous section.

2.3.6 Parallel composition

The process $P \parallel Q$ executes P and Q in lockstep parallel, synchronising on every event. The parallel composition can perform an event if both P and Q can; it can refuse an event if either P or Q can:

$$\mathcal{F}_T\ P \parallel Q\ \rho \mathrel{\widehat=} \{(s, \aleph_P \cup \aleph_Q) \mid (s, \aleph_P) \in \mathcal{F}_T\ P\ \rho \wedge (s, \aleph_Q) \in \mathcal{F}_T\ Q\ \rho\}$$

$P\ {}^X\!\!\parallel^Y Q$ can perform trace s if P can perform the restriction of s to alphabet X, Q can perform the restriction of s to alphabet Y, and all the events of s belong to either X or Y. It can refuse an event from X if P can refuse it; it can refuse an event from Y if Q can refuse it; and it can refuse any events not in X or Y.

$$\begin{aligned}
&\mathcal{F}_T\ P\ {}^X\!\!\parallel^Y Q\ \rho \mathrel{\widehat=} \\
&\quad \{(s, \aleph_P \cup \aleph_Q \cup \aleph_Z) \mid (s\quad X, \aleph_P) \in \mathcal{F}_T\ P\ \rho \wedge (s\quad Y, \aleph_Q) \in \mathcal{F}_T\ Q\ \rho \wedge \Sigma s \subseteq X \cup Y \\
&\qquad\qquad \wedge \Sigma\aleph_P \subseteq X \wedge \Sigma\aleph_Q \subseteq Y \wedge \Sigma\aleph_Z \subseteq \Sigma \setminus X \setminus Y\}
\end{aligned}$$

$P\quad Q$ executes the processes P and Q in parallel without synchronization. It can perform an event if either P or Q can; it can refuse an event if both P and Q can:

$$\mathcal{F}_T\ P\quad Q\ \rho \mathrel{\widehat=} \{(s, \aleph) \mid (s_P, \aleph) \in \mathcal{F}_T\ P\ \rho \wedge (s_Q, \aleph) \in \mathcal{F}_T\ Q\ \rho \wedge s \in s_P\quad s_Q\}$$

where   is defined on traces by

$$s_P\quad s_Q \mathrel{\widehat=} \{s : T\Sigma^*_{\leqslant} \mid \forall t\quad s \uparrow t \cong s_P \uparrow t\quad s_Q \uparrow t\}$$

$P \underset{C}{\parallel} Q$ is a hybrid parallel composition: synchronisation takes place on events from C but all other events are interleaved. An event from C can be performed if both P and Q can perform it, an event from outside C can be performed if either P or Q can perform it. Hence if P can perform trace s_P and Q can perform trace s_Q then $P \underset{C}{\parallel} Q$ can perform any trace from $s_P \underset{C}{\parallel} s_Q$, defined by

$$s_P \underset{C}{\parallel} s_Q \mathrel{\widehat=} \{s \mid s\quad C = s_P\quad C = s_Q\quad C \wedge s \setminus C \in s_P \setminus C\quad s_Q \setminus C\}$$

$P \underset{C}{\|} Q$ can refuse an event from C if either P or Q can refuse it; it can refuse an event from outside C if both P and Q can refuse it.

$$\mathcal{F}_T\ P \underset{C}{\|} Q\ \rho \mathrel{\widehat{=}} \{(s,\aleph) \mid (s_P,\aleph_P) \in \mathcal{F}_T\ P\ \rho \wedge (s_Q,\aleph_Q) \in \mathcal{F}_T\ Q\ \rho \wedge s \in s_P \underset{C}{\|} s_Q$$
$$\wedge\, \aleph \quad C = (\aleph_P \cup \aleph_Q) \quad C \wedge \aleph \setminus C = (\aleph_P \cap \aleph_Q) \setminus C\}$$

2.3.7 Abstraction and renaming

The process $P \setminus X$ acts like P except all events from the set X are made internal. This means:

- events from X occur silently and should not appear in the trace;
- these events do not need the cooperation of the environment; this means that the process P should always be able to perform as many events from X as it requires: this is equivalent to saying that it should be able to refuse any additional events from X.

Thus $P \setminus X$ can perform trace $s \setminus X$ and refuse $\aleph$ if P can perform s and refuse $\aleph \cup ([0, end(s,\aleph)) \times X)$:

$$\mathcal{F}_T\ P \setminus X\ \rho \mathrel{\widehat{=}} \{(s \setminus X, \aleph) \mid (s, \aleph \cup ([0, end(s,\aleph)) \times X)) \in \mathcal{F}_T\ P\ \rho\}$$

The process $f(P)$ acts like P except all events are renamed via the function f. This means:

- $f(P)$ performs the event $f(a)$ if P performs a;
- $f(P)$ can refuse a b if P can refuse all events a such that $f(a) = b$, i.e. if P can refuse $f^{-1}(b)$.

Hence we have the following definition:

$$\mathcal{F}_T\ f(P)\ \rho \mathrel{\widehat{=}} \{(f(s),\aleph) \mid (s, f^{-1}(\aleph)) \in \mathcal{F}_T\ P\ \rho\}$$

The inverse image of P under f may perform a whenever P may perform $f(a)$, and can refuse a whenever P may refuse $f(a)$:

$$\mathcal{F}_T\ f^{-1}(P)\ \rho \mathrel{\widehat{=}} \{(s,\aleph) \mid (f(s), f(\aleph)) \in \mathcal{F}_T\ P\ \rho\}$$

2.3.8 Transfer operators

The process $P \overset{t}{\;} Q$ initially acts like P; if no action is observed by time t then a time out occurs and, after a delay of length δ, control is passed to Q. A behaviour of $P \overset{t}{\;} Q$ is either:

- a behaviour of P whose first event occurs no later than t;
- or a behaviour of P up until time t during which no events occur, followed by a behaviour of Q starting at $t + \delta$:

$$\mathcal{F}_T\ P\ ^{t}\ Q\ \rho \mathrel{\widehat{=}} \{(s,\aleph) \mid begin\ s\quad t \wedge (s,\aleph) \in \mathcal{F}_T\ P\ \rho\}$$
$$\cup\{(s,\aleph) \mid begin\ s\quad t+\delta \wedge (\langle\rangle,\aleph\quad t) \in \mathcal{F}_T\ P\ \rho \wedge (s,\aleph) - t - \delta \in \mathcal{F}_T\ Q\ \rho\}$$

The process $P\ _{t}\ Q$ is similar to $P\ ^{t}\ Q$ except control is removed from P at time t regardless of the progress made. Thus a behaviour of $P\ _{t}\ Q$ must be such that:

- the behaviour up until t is a behaviour of P;
- no events are observed between t and $t+\delta$;
- *and* the behaviour from $t+\delta$ is a behaviour of Q:

$$\mathcal{F}_T\ P\ _{t}\ Q\ \rho \mathrel{\widehat{=}} \{(s,\aleph) \mid (s\quad t,\aleph\quad t) \in \mathcal{F}_T\ P\ \rho \wedge s \uparrow (t,t+\delta) = \langle\rangle \wedge (s,\aleph) - t - \delta \in \mathcal{F}_T\ Q\ \rho\}$$

$P \underset{e}{\nabla} Q$ initially acts like P except it is always willing to perform the interrupt event e; if an e occurs then control is passed to Q. Thus a behaviour of $P \underset{e}{\nabla} Q$ is either:

- a behaviour of P where an e is always available ($e \notin \Sigma\aleph$) but no e occurs ($e \notin \Sigma s$);
- *or* a behaviour of P up until some time t when an e occurs, followed by a behaviour of Q after a delay of δ; in this case an e must not occur before t but must be available up until then:

$$\mathcal{F}_T\ P \underset{e}{\nabla} Q\ \rho \mathrel{\widehat{=}} \{(s,\aleph) \mid e \notin \Sigma(s,\aleph) \wedge (s,\aleph) \in \mathcal{F}_T\ P\ \rho\}$$
$$\cup\{(s,\aleph) \mid \exists t\quad s\quad t\quad e = \langle(t,e)\rangle \wedge e \notin \Sigma(\aleph\quad t) \wedge begin(s\quad t)\quad t+\delta$$
$$\wedge (s\quad t \setminus e, \aleph\quad t) \in \mathcal{F}_T\ P\ \rho \wedge (s,\aleph) - t - \delta \in \mathcal{F}_T\ Q\ \rho\}$$

2.3.9 Recursion

In order to give a semantics to the recursive process $\mu X\quad P$ we need to consider the mapping on the semantic space represented by the term P considered as a function of X. We denote this by $M(X,P)\rho$, defined by

$$M(X,P)\rho \mathrel{\widehat{=}} \lambda Y\quad \mathcal{F}_T\ P\ \rho[Y/X]$$

Recursion is then defined by

$$\mathcal{F}_T\ \mu X\quad P\ \rho \mathrel{\widehat{=}} \text{the unique fixed point of the mapping } M(X,P)\rho$$

In [Dav91] Davies shows that this is well defined if P is constructive for X, where constructivity is defined as follows:

Definition 2.3.2: TCSP term P is *t-constructive* for variable X if

$$\forall t_0 : TIME\ ; \rho : ENV\quad \mathcal{F}_T\ P\ \rho\quad t_0 + t = \mathcal{F}_T\ P\ \rho[\rho\ X\quad t_0/X]\quad t_0 + t$$

◊

Informally, P is t-constructive for X if the behaviour of P up until $t_0 + t$ is independent of the behaviour of X after time t_0.

Definition 2.3.3: Term P is *constructive* for X if there is a strictly positive time t such that P is t-constructive for X. ◇

Davies gives a number of rules for checking whether a term is constructive for a variable.

The recursive process $\mu X \quad P$ differs from $\mu X \quad P$ in that there is a delay of length δ associated with all recursive calls. The mapping on the semantic space associated with P where all calls to X are delayed by δ is denoted by $M_\delta(X, P)\rho$ and defined as follows:

Definition 2.3.4: If P is a TCSP term and X a variable then

$$M_\delta(X, P)\rho \mathrel{\hat=} W_\delta \circ M(X, P)\rho \quad \text{where} \quad W_\delta \mathrel{\hat=} \lambda\, Y \quad \mathcal{F}_T \;\; WAIT\ \delta\, ; X \;\; \rho[Y/X]$$

◇

The function W_δ delays all calls to X by δ. Delayed recursion is defined by

$$\mathcal{F}_T \;\; \mu X \quad P \;\; \rho \mathrel{\hat=} \text{the unique fixed point of the mapping } M_\delta(X, P)\rho$$

In [Ree88] Reed showed that the mapping $M_\delta(X, P)\rho$ is a contraction mapping and so always has a unique fixed point; hence the semantics of $\mu X \quad P$ is well defined.

Mutual recursion is handled similarly. For $i \in I$ let P_i be a term and X_i a variable. We write $\langle X_i = P_i \mid i \in I\rangle_j$ to denote the jth element of the vector of processes $\langle X_i \mid i \in I\rangle$ mutually defined by the set of equations $\{X_i = P_i \mid i \in I\}$. We will write $\underline{P}$ for $\langle P_i \mid i \in I\rangle$, etc. The vector of equations $\langle X_i = P_i\rangle$ represents a mapping on the space S_{TF}^I which contains one copy of S_{TF} for each element of I; this mapping is written $M(\underline{X}, \underline{P})\rho$ and defined by

$$M(\underline{X}, \underline{P})\rho \mathrel{\hat=} \lambda\, \underline{Y} \quad \mathcal{F}_T \;\; \underline{P} \;\; \rho[\underline{Y}/\underline{X}]$$

We then define mutual recursion by

$$\mathcal{F}_T \;\; \langle X_i = P_i\rangle_j \;\; \rho \mathrel{\hat=} S_j \text{ where } \underline{S} \text{ is a fixed point of } M(\underline{X}, \underline{P})\rho$$

In [Dav91] Davies gives a sufficient condition for this to be well defined.

2.4 The proof system

In [DS89b], Davies and Schneider presented a complete proof system for Timed CSP. If P is a Timed CSP term and $S(s, \aleph)$ is a predicate whose free variable represents a behaviour, then they write P **sat**$_\rho$ $S(s, \aleph)$ to specify that all behaviours of P satisfy S:

$$P \ \mathbf{sat}_\rho\ S(s, \aleph) \mathrel{\hat=} \forall (s, \aleph) \in \mathcal{F}_T \;\; P \;\; \rho \quad S(s, \aleph)$$

If P is a process then it makes sense to omit reference to the environment:

$$P \ \mathbf{sat}\ S(s, \aleph) \mathrel{\hat=} \forall (s, \aleph) \in \mathcal{F}_T \;\; P \quad S(s, \aleph)$$

The argument $(s, \aleph)$ is dropped when it is obvious which model we are working in.

They give a proof rule for each construct of the language. These rules are of the following form:

$$\begin{array}{l} \textit{antecedent} \\ \quad\vdots \\ \dfrac{\textit{antecedent}}{\textit{consequent}}\left[\textit{side condition}\right] \end{array}$$

If we can prove each *antecedent* and the *side condition* is true then we can deduce the *consequent*.

On composite processes the proof obligation is reduced to proof obligations on the subcomponents. For example, the proof rule for lockstep parallel composition is

$$\begin{array}{l} P \ \mathrm{sat}_\rho \ S_P \\ Q \ \mathrm{sat}_\rho \ S_Q \\ S_P(s,\aleph_P) \wedge S_Q(s,\aleph_Q) \Rightarrow S(s,\aleph_P \cup \aleph_Q) \\ \hline P \parallel Q \ \mathbf{sat}_\rho \ S \end{array}$$

To prove that $P \parallel Q$ $\mathbf{sat}_\rho$ $S(s,\aleph)$ we have to find specifications S_P and S_Q for P and Q such that whenever behaviours of P and Q satisfy S_P and S_Q the corresponding behaviour of $P \parallel Q$ satisfies S.

Throughout this thesis, we will quote proof rules for the Timed Failures Model as and when we need them.

2.5 The specification language

In order to specify Timed CSP processes, Davies introduced in [Dav91] a specification language; this language was revised in [DRRS93]. The meaning of a specification written in this language is as near as possible to its English language meaning. This means that we can be reasonably confident that specifications written in this language meet our informal requirements. This also means that our specifications will be open to interpretation in other models: in section 5.3 we will present a similar language for specifying prioritized processes, and in section 6.2 we will show that if a Timed CSP process P satisfies a particular specification S written in the specification language, then, subject to certain conditions, all P's prioritized refinements will satisfy S when this is interpreted as a specification on prioritized processes.

2.5.1 Primitive specifications

The predicate $(a \text{ at } t)(s,\aleph)$ specifies that an a occurs at time t:

$$(a \text{ at } t)(s,\aleph) \mathrel{\widehat{=}} \langle (t,a) \rangle \ \mathbf{in} \ s$$

This may be generalised by replacing the event a with a set of events and by replacing the time t with a set of times:

$$A \text{ at } I \mathrel{\widehat{=}} \exists a \in A \ \bullet \ \exists t \in I \ \bullet \ a \text{ at } t$$

A at I holds if some element of A occurs at some time during I. Note that we are using the convention of dropping the argument $(s, \aleph)$ from specifications when it is obvious from the context in which model we are working.

These can be generalised to specify that n events happen during some interval:

$$(A \ \mathsf{at}^n \ I)(s, \aleph) \mathrel{\widehat{=}} \#(s \quad A \uparrow I) \quad n$$

We can also specify that particular events do *not* occur:

$$\begin{aligned} \mathsf{no} \ a \ \mathsf{at} \ t &\mathrel{\widehat{=}} \neg \, (a \ \mathsf{at} \ t) \\ \mathsf{no} \ A \ \mathsf{at} \ I &\mathrel{\widehat{=}} \neg \, (A \ \mathsf{at} \ I) \\ \mathsf{no} \ A \ \mathsf{at}^n \ I &\mathrel{\widehat{=}} \neg \, (A \ \mathsf{at}^n \ I) \end{aligned}$$

Another useful specificatiou primitive is ref, which is used to specify that an event is refused.

$$(a \ \mathsf{ref} \ t)(s, \aleph) \mathrel{\widehat{=}} (t, a) \in \aleph$$

We will not actually write specifications using ref: we will use it to define more useful specification macros.

We can also specify that an event is *not* seen to be refused:

$$\mathsf{no} \ a \ \mathsf{ref} \ t \mathrel{\widehat{=}} \neg \, (a \ \mathsf{ref} \ t)$$

Both of these generalise to a set of events:

$$A \ \mathsf{ref} \ t \mathrel{\widehat{=}} \forall \, a \in A \quad a \ \mathsf{ref} \ t \qquad \mathsf{no} \ A \ \mathsf{ref} \ t \mathrel{\widehat{=}} \forall \, a \in A \quad \mathsf{no} \ a \ \mathsf{ref} \ t$$

We will sometimes want to say that a process acts in a particular way *if* we have observed it for long enough. The predicate beyond t will be true if we have observed it until at least time t:

$$(\mathsf{beyond} \ t)(s, \aleph) \mathrel{\widehat{=}} end(s, \aleph) > t$$

2.5.2 Liveness specifications

The live macro is used to specify that the process is willing to perform an event at a particular time.

$$a \ \mathsf{live} \ t \mathrel{\widehat{=}} a \ \mathsf{at} \ t \lor \mathsf{no} \ a \ \mathsf{ref} \ t$$

a live t is true if either an a is performed at time t or it is not refused. It will be true of an observation of a process if that observation is consistent with the process being able to perform an a at that time.

This can be generalised to take a set of events as argument.

$$A \ \mathsf{live} \ t \mathrel{\widehat{=}} A \ \mathsf{at} \ t \lor \mathsf{no} \ A \ \mathsf{ref} \ t$$

A live t is true if the process is willing to perform any one of the events from A: it will either perform one or refuse none.

We can also generalise the live macro to specify that an event is available throughout some interval, until it is performed:

$$a \text{ live } I \mathrel{\widehat=} \forall t \in I \quad a \text{ at } I \cap [0, t] \vee \text{no } a \text{ ref } t$$

a live I is true if at all times in I, an a cannot be refused unless it has already been observed. This generalises to a set of events in the obvious way:

$$A \text{ live } I \mathrel{\widehat=} \forall t \in I \quad A \text{ at } I \cap [0, t] \vee \text{no } A \text{ ref } t$$

It will be particularly useful to be able to specify that an event becomes available at some time t and remains available until performed.

$$a \text{ live from } t \mathrel{\widehat=} a \text{ live } [t, \infty) \qquad A \text{ live from } t \mathrel{\widehat=} A \text{ live } [t, \infty)$$

Thus from t is simply an abbreviation for the interval $[t, \infty)$.

We can also specify that a process is able to perform up to n copies of an event.

$$\begin{aligned}
a \text{ live}^n\ t &\mathrel{\widehat=} a \text{ at}^n\ t \vee \text{no } a \text{ ref } t \\
A \text{ live}^n\ t &\mathrel{\widehat=} A \text{ at}^n\ t \vee \text{no } A \text{ ref } t \\
a \text{ live}^n\ I &\mathrel{\widehat=} \forall t \in I \quad a \text{ at}^n\ I \cap [0, t] \vee \text{no } a \text{ ref } t \\
A \text{ live}^n\ I &\mathrel{\widehat=} \forall t \in I \quad A \text{ at}^n\ I \cap [0, t] \vee \text{no } A \text{ ref } t
\end{aligned}$$

2.5.3 History predicates

Often we will want to write specifications that refer in some way to the events that have been observed. These will take the form $\varphi(M(s))$, where M is a projection function from timed traces to some type T, and φ is a predicate on T. We can define a few useful such projection functions M.

The functions first and last return the first or last timed events observed during a behaviour:

$$\text{first}(s) \mathrel{\widehat=} \textit{head } s \qquad \text{last}(s) \mathrel{\widehat=} \textit{foot } s$$

These can be qualified with one of the terms before t, after t or during I to restrict attention to a particular set of times. We can also restrict our attention to a particular set of events. For example:

$$\begin{aligned}
(\text{first } A \text{ after } t)(s) &\mathrel{\widehat=} \textit{head}(s \quad A \quad t) \\
(\text{last } A \text{ before } t)(s) &\mathrel{\widehat=} \textit{foot}(s \quad A \quad t) \\
(\text{last during } I)(s) &\mathrel{\widehat=} \textit{foot}(s \uparrow I)
\end{aligned}$$

The functions time of and name of return the time and event components of a timed event:

$$\text{time of } (t, a) \mathrel{\widehat=} t \qquad \text{name of } (t, a) \mathrel{\widehat=} a$$

These can be used to write predicates such as

$$\text{time of first } A \text{ after } 2 \quad 3 \qquad \text{name of last } A = a$$

Other functions that we will find useful are alphabet which returns the set of (untimed) events observed, and count A which returns the number of events from the set A that are performed:

$$\mathsf{alphabet}(s) \mathrel{\widehat{=}} \Sigma s \qquad \mathsf{count}\, A(s) \mathrel{\widehat{=}} \#(s \upharpoonright A)$$

These can be qualified with the phrases before t, after t or during I; we will omit the argument A of count if we want to refer to the total number of events performed, i.e. in the case $A = \Sigma$.

2.5.4 Environmental assumptions

Often we will want to say that a process acts in a particular way *if* the environment satisfies some condition. In this subsection we describe a few macros for placing conditions on the environment.

We will write a open t to specify that the environment is willing to perform an a at time t:

$$a\ \mathsf{open}\ t \mathrel{\widehat{=}} a\ \mathsf{at}\ t \vee a\ \mathsf{ref}\ t$$

a open t is true if the observation is consistent with the environment being willing to perform an a at time t: it is true if an a is either performed or refused at time t.

This can be extended to sets of events in the obvious way:

$$A\ \mathsf{open}\ t \mathrel{\widehat{=}} A\ \mathsf{at}\ t \vee A\ \mathsf{ref}\ t$$

We will say a open I if the environment is willing to perform an a at all times during I until one is performed:

$$\begin{aligned} a\ \mathsf{open}\ I &\mathrel{\widehat{=}} \forall t \in I \;\; a\ \mathsf{at}\ I \cap [0, t] \vee a\ \mathsf{ref}\ t \\ A\ \mathsf{open}\ I &\mathrel{\widehat{=}} \forall t \in I \;\; A\ \mathsf{at}\ I \cap [0, t] \vee A\ \mathsf{ref}\ t \end{aligned}$$

As with live, it is useful to have a special form for the interval $[t, \infty)$:

$$\begin{aligned} a\ \mathsf{open\ from}\ t &\mathrel{\widehat{=}} a\ \mathsf{open}\ [t, \infty) \\ A\ \mathsf{open\ from}\ t &\mathrel{\widehat{=}} A\ \mathsf{open}\ [t, \infty) \end{aligned}$$

It is also useful to be able to generalise to say that the environment is able to perform n copies of an event:

$$\begin{aligned} a\ \mathsf{open}^n\ t &\mathrel{\widehat{=}} a\ \mathsf{at}^n\ t \vee a\ \mathsf{ref}\ t \\ A\ \mathsf{open}^n\ t &\mathrel{\widehat{=}} A\ \mathsf{at}^n\ t \vee A\ \mathsf{ref}\ t \\ a\ \mathsf{open}^n\ I &\mathrel{\widehat{=}} \forall t \in I \;\; a\ \mathsf{at}^n\ I \cap [0, t] \vee a\ \mathsf{ref}\ t \\ A\ \mathsf{open}^n\ I &\mathrel{\widehat{=}} \forall t \in I \;\; A\ \mathsf{at}^n\ I \cap [0, t] \vee A\ \mathsf{ref}\ t \end{aligned}$$

The following lemma shows that the open macro does what we want:

Lemma 2.5.1: $A\ \mathsf{open}\ t \wedge A\ \mathsf{live}\ t \Rightarrow A\ \mathsf{at}\ t$. ♡

If the environment is willing to perform any event from A and the process is live on A, then an event from A occurs.

Proof: We have

$$\begin{array}{ll} & A \text{ open } t \wedge A \text{ live } t \\ \Rightarrow & \langle \text{definitions} \rangle \\ & (A \text{ at } t \vee \forall a \in A \bullet a \text{ ref } t) \wedge (A \text{ at } t \vee \forall a \in A \bullet \neg\, a \text{ ref } t) \\ \Rightarrow & \langle \text{predicate calculus} \rangle \\ & A \text{ at } t \end{array}$$

□

To specify that the environment is *not* willing to perform an event, we use the closed macro:

$$a \text{ closed } t \mathrel{\widehat{=}} \neg\, (a \text{ at } t)$$

If a closed t holds then the observation is consistent with the environment being unwilling to perform an a at time t. Note that this is the same as no a at t: we will restrict the use of closed to environmental assumptions. This macro generalises in the obvious way:

$$A \text{ closed } I \mathrel{\widehat{=}} \forall a \in A \bullet \forall t \in I \bullet a \text{ closed } t$$

The final environmental assumption we want is to say that the environment is *always* willing to perform as many events from a set A as the process wants. This will occur when the events from A are hidden.

$$\text{internal } A \mathrel{\widehat{=}} \forall t \bullet \text{beyond } t \Rightarrow A \text{ ref } t$$

Note that

$$(\text{internal } A)(s, \aleph) = A \text{ open}^{\infty} [0, end(s, \aleph))$$

2.6 Recent changes

The above description of Timed CSP follows mainly that described in [Dav91], although the specification language is that of [DRRS93]. Recently a couple of small changes have been made to the semantics [DS92a]; for completeness, we include here a note of these changes, although the new models presented in this thesis will be based upon the earlier work.

In the earlier models there was a non-zero lower bound δ between the times at which causally related events could occur. More recently, this constraint has been dropped, and a prefixing operator with a zero delay has been introduced. For example, the process

$$a \xrightarrow{0} b \xrightarrow{1} SKIP$$

May perform an a and a b at the same instant, and then terminate one second later. For example, it may perform the trace $\langle (0, a), (0, b) \rangle$. In order to incorporate immediate prefixing into the semantic model, it was necessary to drop axiom 3, which allowed simultaneous events to be reordered. If this axiom were retained, then the above process would be able to perform the trace $\langle (0, b), (0, a) \rangle$, and so by axiom 2 would also be able to perform the trace $\langle (0, b) \rangle$, which is obviously nonsense.

The other main change that has been made to the semantic definitions is that now a distributed system may terminate only when all components can terminate. Thus in the parallel combinations

$$P \; {}_{A}\|_{B} \; Q \quad \text{and} \quad P \underset{C}{\|} Q$$

the event   is implicitly included in the synchronization set, and the interleaving operator may be defined by the equation

$$P \quad Q = P \underset{\checkmark}{\|} Q$$

Chapter 3

The Prioritized Model

In this chapter we present the syntax and semantics of the prioritized language. Recall that, as described in the introduction, one of our aims is to restrict nondeterminism to just that caused by the nondeterministic choice operator, so that when we replace the nondeterministic choice operator by a probabilistic choice operator, we will be able to present a semantic model that gives the probability of a process acting in a certain way.

In section 3.1 we describe the syntax of the language. In section 3.2 we illustrate the language with a couple of examples. We describe the semantic space in section 3.3 and give semantic definitions for all the constructs of the language in section 3.4. In section 3.5 we describe how the semantic model can be extended to model communication of values over channels. In section 3.6 we show that by removing the nondeterministic choice operator from the syntax, we are left with a language that is completely deterministic.

3.1 Syntax for the prioritized language

We want to produce a language where the only form of nondeterminism is that caused by the nondeterministic choice operator. In order to do this we must first understand the ways in which nondeterminism can arise. Nondeterminism can arise in Timed CSP in a number of ways:

Explicit nondeterminism: The process $P \sqcap Q$ chooses nondeterministically between the processes P and Q.

External choice: Consider the process $a \longrightarrow P \quad b \longrightarrow Q$. If the environment is willing to do either an a or a b at some time, then the choice is made nondeterministically.

Interleaving: Consider the process $a \longrightarrow P \quad b \longrightarrow Q$. If the environment is willing to perform either an a or a b at some time (but not both), then the choice is made nondeterministically.

Hiding and renaming: Deterministic processes can sometimes be made nondeterministic by hiding or renaming. For example, if the process $a \longrightarrow P \quad b \longrightarrow Q$ is put in an environment that offers just a b at time 0, then the b will be performed. If however the process $(a \longrightarrow P \quad b \longrightarrow Q) \setminus a$ is put in the same environment then it will nondeterministically choose between performing the b or performing the a silently.

The last three forms can all be thought of as types of underspecification; in normal Timed CSP we do not specify how the operators behave in the situations described. We shall refine our operators so as to overcome this underspecification.

3.1.1 Biased external choice

We define two operators, a left-biased[1] choice $\overleftarrow{\square}$, and a right-biased choice $\overrightarrow{\square}$. The left-biased choice $P \overleftarrow{\square} Q$ will choose P if the environment is willing to do the first events of both P and Q (at some time). The right-biased choice $P \overrightarrow{\square} Q$ will choose Q if the environment is willing to do the first events of both P and Q. For example, a customer who is willing to accept a toffee, but would prefer a chocolate:

$$CUST \mathrel{\widehat{=}} \mathit{chocolate} \overleftarrow{\square} \mathit{toffee}$$

where we have written *chocolate* as an abbreviation for $\mathit{chocolate} \longrightarrow STOP$.

3.1.2 Parallel composition

Consider the process $(a \overleftarrow{\square} b) \parallel (a \overrightarrow{\square} b)$. If the environment offers both a and b at time 0, then the behaviour of the process is not fully specified. The left hand side wants to perform an a, while the right hand side wants to perform a b. The only sensible interpretation is that the process chooses nondeterministically between the a and the b. Since we are aiming to eliminate all nondeterminism, we define a left biased parallel operator $\overleftarrow{\parallel}$ which arbitrates in favour of its left hand argument. So $(a \overleftarrow{\square} b) \overleftarrow{\parallel} (a \overrightarrow{\square} b)$ will perform an a if the environment offers both a and b. We can consider the left hand side to be a master, and the right hand side to be a slave which will do whatever its master wants, if it can.

For example, consider a vending machine which will dispense either chocolates or toffees as its environment requires, but would rather dispense toffees:

$$VMB \mathrel{\widehat{=}} \mathit{chocolate} \overrightarrow{\square} \mathit{toffee}$$

If we put this in parallel with the customer who prefers chocolates, with the customer acting as the master, then the customer gets what he wants:

$$CUST \overleftarrow{\parallel} VMB = \mathit{chocolate} \overleftarrow{\square} \mathit{toffee}$$

If however we make the machine the master, then it gets its way:

$$VMB \overleftarrow{\parallel} CUST = \mathit{chocolate} \overrightarrow{\square} \mathit{toffee}$$

We can similarly define a right biased parallel operator $\overrightarrow{\parallel}$ which arbitrates in favour of its right hand argument. For example, $(a \overleftarrow{\square} b) \overrightarrow{\parallel} (a \overrightarrow{\square} b)$ will perform a b if the environment offers both an a and a b.

[1]Throughout this thesis we will use the words *biased* and *prioritized* as synonyms.

3.1.3 Interleaving

We define a left biased interleave operator $\longleftarrow$ such that if the environment is willing to do events of P or of Q (but not both) then $P \longleftarrow Q$ performs the events of P. For example:

- if a single a is offered then $a \longrightarrow P \longleftarrow a \longrightarrow Q$ will perform the a on the left;
- $(a \longleftarrow b) \mathbin{\overleftarrow{\parallel}} (a \mathbin{\overleftarrow{\Box}} b)$ will perform an a if an a and a b are offered at the same time.
- $(a \longleftarrow b) \mathbin{\overrightarrow{\parallel}} (a \mathbin{\overrightarrow{\Box}} b)$ will perform a b if an a and a b are offered at the same time, since the right hand side is the master and it prefers the b.
- A greedy customer would like both a chocolate and a toffee, but if he can have only one he would prefer a chocolate:

  $$GCUST \mathrel{\widehat{=}} chocolate \longleftarrow toffee$$

  When he is placed in parallel with the biased vending machine, with him as the master, he gets just a chocolate since the vending machine is only willing to dispense one sweet.

  $$GCUST \mathbin{\overleftarrow{\parallel}} VMB = chocolate \mathbin{\overleftarrow{\Box}} toffee$$

We can similarly define a right biased interleave operator $\longrightarrow$ such that if the environment is willing to do events of P or of Q (but not both) then $P \longrightarrow Q$ performs the events of Q.

Aside: The reader may be wondering why we have not specified that if processes P and Q have different initial events then $P \longleftarrow Q$ offers these events equally strongly, and allows the environment to decide which is performed. This method does not work, as can be seen by considering the process $(a \longleftarrow (b \mathbin{\overleftarrow{\Box}} c)) \mathbin{\overleftarrow{\parallel}} (c \mathbin{\overleftarrow{\Box}} a \mathbin{\overleftarrow{\Box}} b)$. Suppose this process is offered both an a and a b; then the left hand side has no preference between them, and so the right hand side chooses a. Similarly, if it is offered an a and a c, the right hand side makes the choice in favour of c. If, however, it is offered a b and a c, then the left hand side chooses in favour of b. So this process prefers a to b, prefers b to c and prefers c to a. We conclude that it is not possible to define the interleave operator in this way.

3.1.4 Alphabet parallel composition

The ideas of the previous sections carry over to the parameterized parallel operators. The priorities of $P \;{}_{A}\!\!\mathbin{\overleftarrow{\parallel}}_{B}\; Q$ follow the priorities of P on events from A, and follow the priorities of Q on events from $B \setminus A$; events from the master's alphabet (A) are preferred to other events (those from $B \setminus A$). The priorities of $P \;{}_{A}\!\!\mathbin{\overrightarrow{\parallel}}_{B}\; Q$ follow the priorities of Q on events from B, and follow the priorities of P on events from $A \setminus B$.

$P \mathbin{\overleftarrow{\parallel}}_{C} Q$ and $P \mathbin{\overrightarrow{\parallel}}_{C} Q$ execute P and Q in parallel, synchronising on events in C. They are biased towards P and Q respectively.

3.1.5 Complete syntax

The complete syntax for Biased Timed CSP (BTCSP) is as follows

$$
\begin{array}{rll}
P ::= & STOP \mid SKIP \mid WAIT\ t \mid X \mid & \text{basic processes} \\
& a \xrightarrow{t} P \mid P \quad P \mid WAIT\ t\,;P \mid & \text{sequential composition} \\
& P \sqcap P \mid \quad {}_{i \in I} P_i \mid P \,\square\, P \mid P \,\square\, P \mid & \text{alternation} \\
& P \mathbin{\leftarrow\!\!\!\|} P \mid P \mathbin{\|\!\!\!\rightarrow} P \mid P \;{}^{A}\!\!\mathbin{\leftarrow\!\!\!\|}{}^{B}\; P \mid P \;{}^{A}\!\!\mathbin{\|\!\!\!\rightarrow}{}^{B}\; P \mid & \text{parallel composition} \\
& P \longleftarrow P \mid P \longrightarrow P \mid P \underset{A}{\leftarrow\!\!\!\|} P \mid P \underset{A}{\|\!\!\!\rightarrow} P \mid & \text{interleaving} \\
& P \overset{t}{\quad} P \mid P \underset{t}{\quad} P \mid P \underset{a}{\nabla} P \mid & \text{transfer operators} \\
& P \setminus A \mid f(P) \mid & \text{abstraction and renaming} \\
& \mu X \quad P \mid \mu X \quad P \mid \langle X_i = P_i \rangle_j & \text{recursion}
\end{array}
$$

where t ranges over the set $TIME$ of times, which we take to be positive real numbers; X ranges over the space VAR of variables; a ranges over some alphabet Σ of events; A and B range over Σ; f ranges over $\Sigma \to \Sigma$; and i and j range over an indexing set I.

3.1.6 The effect of hiding

Consider the process $P \mathrel{\widehat{=}} (a \,\square\, b) \setminus a$. It is interesting to ask whether this process can ever perform a b. The process P certainly prefers to perform an a (silently) to a b. In a previous paper [Low91a] we took the view that the environment would always be willing to perform the empty bag of events; hence P could never perform a b since it would always choose to perform a silent a in preference. This assumption produces a model which, while sound, is extremely complicated and contains a number of unusual and undesirable features.

In this thesis we adopt the view that there are environments that are *not* willing to idle. Then the process P *is* able to perform a b, but only if its environment is not willing to perform the empty bag of events. Consider for example the process $b \mathbin{\leftarrow\!\!\!\|} P$. The left hand side of this prefers to perform a b than to idle; it is the master and so it forces P to perform the b even though it would prefer to perform a silent a.

3.2 Examples: a lift system and an interrupt mechanism

In this section we consider two examples that make use of the biased operators.

3.2.1 A lift mechanism

We consider an example of a lift serving three floors of a building: on each floor there is a button that can be used to summon the lift; once the button has been pressed, the lift should arrive on that floor after a short delay. The naïve implementation in unprioritized Timed CSP would be

$$SYSTEM \mathrel{\widehat{=}} (LIFT\ {}^{A \cup R}\|^{R \cup P}\ BUTTONS) \setminus R$$

$$\begin{aligned}
LIFT &\mathrel{\widehat{=}} req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT \\
&\qquad req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT \\
&\qquad req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT \\
BUTTONS &\mathrel{\widehat{=}} BUTTON_0 \quad BUTTON_1 \quad BUTTON_2 \\
BUTTON_i &\mathrel{\widehat{=}} push_i \xrightarrow{1} req_i \xrightarrow{1} BUTTON_i \qquad (i = 0, 1, 2)
\end{aligned}$$

where the alphabets are defined by

$$A \mathrel{\widehat{=}} \{arrive_i \mid i \in 0 \,..\, 2\} \qquad R \mathrel{\widehat{=}} \{req_i \mid i \in 0 \,..\, 2\} \qquad P \mathrel{\widehat{=}} \{push_i \mid i \in 0 \,..\, 2\}$$

When button i is pushed, it makes a request to the lift by offering the event req_i; two seconds after the req_i is accepted, the lift arrives at floor i.

Unfortunately, there is a problem with this implementation. Suppose you are on the first floor and the lift is on the ground floor. You press your button at the same moment that somebody on the second floor presses the button there. Both buttons offer their *req* event, and suppose the lift chooses in favour of the button on the second floor; then the lift goes straight past you to arrive on the second floor. Meanwhile, somebody arrives on the ground floor and pushes the button there. The buttons on the first and ground floors are now both offering their *req* events; suppose the lift chooses in favour of the one on the ground floor; again, the lift goes straight past you, to reach the ground floor. This frustrating sequence of events could continue until you eventually give up and head for the stairs.

This is not the only problem. There is also the possibility that you are stuck on the second floor while the lift shuttles backwards and forwards between the ground and first floor. It's even possible that the lift never leaves the ground floor, if more and more people keep on pressing the button there.

These problems can be overcome using biased operators. We use the following definitions:

$$\begin{aligned}
SYSTEM &\mathrel{\widehat{=}} (LIFT\ {}^{A\cup R}\overleftarrow{\|}{}^{R\cup P}\ BUTTONS) \setminus R \\
LIFT &\mathrel{\widehat{=}} LIFT_0 \\
LIFT_0 &\mathrel{\widehat{=}} req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\uparrow} \\
&\qquad [\!\Box\ req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\
&\qquad [\!\Box\ req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\
LIFT_1^{\uparrow} &\mathrel{\widehat{=}} req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\
&\qquad [\!\Box\ req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\
&\qquad [\!\Box\ req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\uparrow} \\
LIFT_1^{\downarrow} &\mathrel{\widehat{=}} req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\
&\qquad [\!\Box\ req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\
&\qquad [\!\Box\ req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\downarrow} \\
LIFT_2 &\mathrel{\widehat{=}} req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\downarrow} \\
&\qquad [\!\Box\ req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\
&\qquad [\!\Box\ req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\
BUTTONS &\mathrel{\widehat{=}} BUTTON_0 \quad BUTTON_1 \quad BUTTON_2 \\
BUTTON_i &\mathrel{\widehat{=}} push_i \xrightarrow{1} req_i \xrightarrow{1} BUTTON_i \qquad (i = 0, 1, 2)
\end{aligned}$$

where the interleaving of the buttons could be either left- or right-biased. $LIFT_0$ and $LIFT_2$ represent the lift on the ground and second floors respectively; $LIFT_1^{\uparrow}$ and $LIFT_1^{\downarrow}$ represent the lift on the first floor where the previous movement was up or down respectively. The lift is biased in favour of next going to an adjacent floor; when it is on the first floor it is biased in favour of continuing in the direction it last went. The reader may care to verify that none of the problems described above occur given these definitions.

In section 5.5 we will formally verify that if the environment always allows the *arrive* events then the lift arrives at a floor within 15 seconds of the button being pressed.

3.2.2 An interrupt mechanism

We consider now an example of an interrupt mechanism, introduced in [CH88], and illustrated in figure 3.1. A counter can normally continually perform the events *up* and *down*. If,

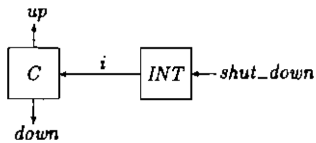

Figure 3.1: A counter with interrupt mechanism

however, the event *shut_down* occurs, then it should be interrupted via the internal event i. In an unprioritized model the definition would be

$$\begin{aligned}
SYS &\mathrel{\widehat{=}} (C_0 \; {}^{X}\|^{Y} \; INT) \setminus i \\
INT &\mathrel{\widehat{=}} shut_down \longrightarrow i \longrightarrow STOP \\
C_0 &\mathrel{\widehat{=}} up \longrightarrow C_1 \quad i \longrightarrow STOP \\
C_{n+1} &\mathrel{\widehat{=}} (up \longrightarrow C_{n+2} \quad down \longrightarrow C_n) \quad i \longrightarrow STOP
\end{aligned}$$

where the alphabets are given by $X \mathrel{\widehat{=}} \{up, down, i\}$, $Y \mathrel{\widehat{=}} \{i, shut\_down\}$.

It should be obvious that this could perform the trace $\langle up, down, shut\_down, up, down \rangle$: C can choose to ignore the event i, offered by INT, in favour of *ups* and *downs*. We can get around this by giving the i a higher priority than the *up* and the *down*:

$$\begin{aligned}
C_0 &\mathrel{\widehat{=}} up \longrightarrow C_1 \sqsupset i \longrightarrow STOP \\
C_{n+1} &\mathrel{\widehat{=}} (up \longrightarrow C_{n+2} \quad down \longrightarrow C_n) \sqsupset i \longrightarrow STOP
\end{aligned}$$

where the external choice could be either left- or right-biased. Now the i will be performed as soon as it is offered, and C will be interrupted as required.

3.3 The semantic model

In this section we develop a semantic model for our language. We begin by describing how we want to model a behaviour of a process. We then present some notation before producing the semantic model itself, which will represent a process by the set of behaviours that it can perform.

3.3.1 Behaviours

As in most models of concurrency, we want our model of a behaviour, or observation, of a process to record the events performed. Since we are interested in the different priorities given to different actions, we also want to include some representation of these priorities. It will ease our notation to also include the time at which the observation ends. Our model of a behaviour will therefore consist of three parts: the time up until which the process is observed, the events which it performs and the priorities given to different actions.

The trace of a process is the collection of timed events which it performs. In standard Timed CSP the traces $\langle(0,a),(0,b)\rangle$ and $\langle(0,b),(0,a)\rangle$ are treated as distinct. In this thesis we want to associate these, otherwise when we come to consider probabilities we will experience problems. For example, consider the process $a \longleftarrow b$; if the environment is willing to perform an a and a b at time 0 then this can perform the trace $\langle(0,a),(0,b)\rangle$ with probability one and can also perform the trace $\langle(0,b),(0,a)\rangle$ with probability one: our probabilities will not sum to one. In our model we shall say that in this environment the process performs the bag $\{\!|a,b|\!\}$ at time 0 with probability one.

We represent traces as functions from an initial segment of the time domain to bags of events:

Definition 3.3.1 (Timed traces) The space TT of timed traces is defined by

$$TT \mathrel{\hat{=}} \{s : TIME \twoheadrightarrow \operatorname{bag}\Sigma \mid \exists \tau \quad \operatorname{dom} s = [0,\tau]\}$$

$\diamond$

We think of $s(t)$ as being the bag of events performed at time t. Both of the above traces are represented by $\lambda t \quad \text{if } t = 1 \text{ then } \{\!|a,b|\!\} \text{ else } \{\!||\!\}$. For ease of notation, we shall often write traces as sequences within the brackets $\prec$ and $\succ$, so the above trace will be denoted by either $\prec(1,a),(1,b)\succ$ or $\prec(1,b),(1,a)\succ$, and the empty trace is written $\prec\succ$. We shall sometimes omit the brackets for singleton traces.

We say that a process offers a particular bag of events if it is willing to perform that bag, or, put another way, if it offers the bag to parallel processes.

Definition 3.3.2 (Offers) The set of offers OFF is defined by $OFF \mathrel{\hat{=}} TIME \times \operatorname{bag}\Sigma$.

$\diamond$

The pair (t,χ) represents the bag of events χ being offered at time t. We shall write v, w, etc. for typical members of OFF, and χ, ψ, etc. for typical members of $\operatorname{bag}\Sigma$.

Note: It is normal to consider a function from type a to type b to be of type $\mathbb{P}(a \times b)$. Using this identification, we can consider a timed trace to be of type $\mathbb{P}(TIME \times \operatorname{bag}\Sigma)$, i.e. a trace is simply a collection of offers. We will make use of this to simplify our notation.

A process will often be willing to offer more than one particular bag of events. It will then have some preference as to which bag of events it would rather perform. For example, the process $a \longleftarrow b$ initially offers the bags $\{\!|a,b|\!\}$, $\{\!|a|\!\}$, $\{\!|b|\!\}$, and $\{\!||\!\}$, and prefers $\{\!|a,b|\!\}$ to $\{\!|a|\!\}$, prefers $\{\!|a|\!\}$ to $\{\!|b|\!\}$, and prefers $\{\!|b|\!\}$ to $\{\!||\!\}$. We want to model the order of preference of offers.

Definition 3.3.3 (Offer relations) We define the space $OFFREL$ of offer relations to be those relations $\sqsubseteq$ of type $OFF \times OFF$ satisfying the following conditions:

1. $(t,\chi) \sqsubseteq (t',\chi') \Rightarrow t = t'$ (comparable offers occur at the same time)
2. $w \sqsubseteq w' \wedge w' \sqsubseteq w'' \Rightarrow w \sqsubseteq w''$ (transitivity)
3. $w \sqsubseteq w' \wedge w' \sqsubseteq w \Rightarrow w = w'$ (antisymmetry)
4. $w \in \text{items} \sqsubseteq \Rightarrow w \sqsubseteq w$ (reflexivity on items $\sqsubseteq$)
5. $(t,\chi),(t,\psi) \in \text{items} \sqsubseteq \Rightarrow (t,\chi) \sqsubseteq (t,\psi) \vee (t,\psi) \sqsubseteq (t,\chi)$ (totality on items $\sqsubseteq$)

where items $\sqsubseteq$ is the set of all offers made by the process:

$$\text{items} \sqsubseteq \mathrel{\widehat{=}} \{w \mid \exists\, v \quad w \sqsubseteq v \vee v \sqsubseteq w\}$$

◇

Informally, if $v \sqsubseteq w$ then the process would rather perform w than v. For example, $a \longleftarrow b$ has offer relation with $(0, \{\!|\,|\!\}) \sqsubseteq (0, \{\!|b|\!\}) \sqsubseteq (0, \{\!|a|\!\}) \sqsubseteq (0, \{\!|a, b|\!\})$.

Note in particular condition 5 which says that the restriction of an offer relation to a particular instant is a total order on those offers that the process is willing to perform.

We introduce the following shorthands:

$$v \sqsubset w \Leftrightarrow v \sqsubseteq w \wedge v \neq w \qquad v \sqsupseteq w \Leftrightarrow w \sqsubseteq v \qquad v \sqsupset w \Leftrightarrow w \sqsubset v$$

A behaviour will be a triple of type $TIME \times OFFREL \times TT$. The behaviour $(\tau, \sqsubseteq, s)$ will represent an observation up until time τ where trace s is observed and where $\sqsubseteq$ gives the priorities on offers. We shall discuss which behaviours are possible after we have introduced some notation.

An environmental offer is the set of bags of timed events which the process is offered by the environment; more formally, it is a set of offers, i.e. a set of type $\quad(OFF)$. We let $EOFF$ be the set of all environmental offers and write Ω for a typical member. We shall discuss environmental offers more fully after we have introduced some notation.

3.3.2 Notation

Our notation is based upon the notation for the Timed Failures Model, described in section 2.2.2. An index of notation appears on pages 214–218.

Our notation for bags follows that of Morgan [Mor90]. We write $b.e$ for the number of times element e occurs in bag b; $e \in b$ is true iff $b.e > 0$. We have a number of operations on bags

$$(b_1 \cup b_2).e = b_1.e \sqcup b_2.e \qquad (b_1 \cap b_2).e = b_1.e \sqcap b_2.e$$
$$(b_1 - b_2).e = (b_1.e - b_2.e) \sqcup 0 \qquad (b_1 \uplus b_2).e = b_1.e + b_2.e$$

where the operators $\sqcup$ and $\sqcap$ return the maximum and minimum of their arguments respectively. Bag enumerations and bag comprehensions are written within bag brackets $\{\!|$ and $|\!\}$. If a particular value of a bound variable occurs more than once in a bag comprehension, then the corresponding term occurs more than once.

The function *times* returns the set of times at which events occur during a trace:

$$times\ s \mathrel{\widehat{=}} \{t \mid s(t) \neq \{\!|\,|\!\}\}$$

This contrasts with the function I which returns the set of all times in the domain of a trace:

$$Is \mathrel{\widehat=} \operatorname{dom} s$$

We can define similar functions for offers, offer relations, and environmental offers:

$$\begin{aligned}
I(t,\chi) &\mathrel{\widehat=} t \\
I\sqsubseteq &\mathrel{\widehat=} \{t \mid \exists \chi \quad (t,\chi) \in \text{items} \sqsubseteq\} \\
\mathit{times}\,\Omega &\mathrel{\widehat=} \{t \mid \exists \chi \neq \{\!|\,|\!\} \quad (t,\chi) \in \Omega\} \\
I\Omega &\mathrel{\widehat=} \{t \mid \exists \chi \quad (t,\chi) \in \Omega\}
\end{aligned}$$

We will consider only those offer relations $\sqsubseteq$ and environmental offers Ω such that $I\sqsubseteq$ and $I\Omega$ are intervals.

We define *begin* and *end* operators which return the times of the first and last events of a trace:

$$\mathit{begin}\,s \mathrel{\widehat=} \begin{cases} \infty & \text{if } \mathit{times}\,s = \{\} \\ \inf(\mathit{times}\,s) & \text{otherwise} \end{cases} \qquad \mathit{end}\,s \mathrel{\widehat=} \begin{cases} 0 & \text{if } \mathit{times}\,s = \{\} \\ \sup(\mathit{times}\,s) & \text{otherwise} \end{cases}$$

It will also be useful to define the function *begin* on environmental offers: it will return the time at which the environment is first willing to perform an event:

$$\mathit{begin}\,\Omega \mathrel{\widehat=} \begin{cases} \infty & \text{if } \mathit{times}\,\Omega = \{\} \\ \inf(\mathit{times}\,\Omega) & \text{if } \mathit{times}\,\Omega \neq \{\} \end{cases}$$

The *first* and *last* operators return the bags of initial or final events of a non-empty trace:

$$\mathit{first}\,s \mathrel{\widehat=} s(\mathit{begin}\,s) \qquad \mathit{last}\,s \mathrel{\widehat=} s(\mathit{end}\,s)$$

The *head* and *foot* operators return the first and last non-empty offers performed:

$$\mathit{head}\,s \mathrel{\widehat=} (\mathit{begin}\,s, \mathit{first}\,s) \qquad \mathit{foot}\,s \mathrel{\widehat=} (\mathit{end}\,s, \mathit{last}\,s)$$

The during operator $\uparrow$ returns the subtrace of a trace that occurs during some time interval:

$$s \uparrow I \mathrel{\widehat=} \{t \mapsto s(t) \mid t \in I\}$$

We can define similar operators on offer relations and environmental offers:

$$\begin{aligned}
\sqsubseteq \uparrow I &\mathrel{\widehat=} \sqsubseteq' \quad \text{where } (t,\chi) \sqsubseteq' (t,\psi) \Leftrightarrow t \in I \wedge (t,\chi) \sqsubseteq (t,\psi) \\
\Omega \uparrow I &\mathrel{\widehat=} \{(t,\chi) \in \Omega \mid t \in I\}
\end{aligned}$$

We use these to define before ( ), strictly before ( ), after ( ), strictly after ( ) and at ($\uparrow$) operators:

$$\begin{array}{lll}
s \quad t \mathrel{\widehat=} s \uparrow [0,t] & \sqsubseteq \quad t \mathrel{\widehat=} \sqsubseteq \uparrow [0,t] & \Omega \quad t \mathrel{\widehat=} \Omega \uparrow [0,t] \\
s \quad t \mathrel{\widehat=} s \uparrow [0,t) & \sqsubseteq \quad t \mathrel{\widehat=} \sqsubseteq \uparrow [0,t) & \Omega \quad t \mathrel{\widehat=} \Omega \uparrow [0,t) \\
s \quad t \mathrel{\widehat=} s \uparrow [t,\infty) & \sqsubseteq \quad t \mathrel{\widehat=} \sqsubseteq \uparrow [t,\infty) & \Omega \quad t \mathrel{\widehat=} \Omega \uparrow [t,\infty) \\
s \quad t \mathrel{\widehat=} s \uparrow (t,\infty) & \sqsubseteq \quad t \mathrel{\widehat=} \sqsubseteq \uparrow (t,\infty) & \Omega \quad t \mathrel{\widehat=} \Omega \uparrow (t,\infty) \\
s \uparrow t \mathrel{\widehat=} (t, s(t)) & \sqsubseteq \uparrow t \mathrel{\widehat=} \sqsubseteq \uparrow \{t\} & \Omega \uparrow t \mathrel{\widehat=} \Omega \uparrow \{t\}
\end{array}$$

We define a partial concatenation operator on traces:

$$s_1 \quad s_2 = \{t \mapsto s_1(t) \mid t \in Is_1\} \cup \{t \mapsto s_2(t) \mid t \in Is_2\}$$
$$\text{if } \exists \tau \quad Is_1 \cup Is_2 = [0, \tau] \wedge \forall t_1 \in Is_1 ; t_2 \in Is_2 \quad t_1 < t_2$$

This is only defiued if the time intervals of s_1 and s_2 follow one another without a gap and without overlap. We define similar operations on offer relations and environmental offers:

$$\sqsubseteq_1 \quad \sqsubseteq_2 \;\widehat{=}\; \sqsubseteq_1 \cup \sqsubseteq_2 \quad \text{if } \exists \tau \quad I\sqsubseteq_1 \cup I\sqsubseteq_2 = [0, \tau] \wedge \forall t_1 \in I\sqsubseteq_1 ; t_2 \in I\sqsubseteq_2 \quad t_1 < t_2$$
$$\Omega_1 \quad \Omega_2 \;\widehat{=}\; \Omega_1 \cup \Omega_2 \quad \text{if } \exists \tau \quad I\Omega_1 \cup I\Omega_2 = [0, \tau] \wedge \forall t_1 \in I\Omega_1 ; t_2 \in I\Omega_2 \quad t_1 < t_2$$

We define restriction and hiding operators on offers, traces, and environmental offers:

$$(t, \chi) \quad X \;\widehat{=}\; (t, \{\!|a \in \chi \mid a \in X|\!\}) \qquad (t, \chi) \setminus X \;\widehat{=}\; (t, \{\!|a \in \chi \mid a \notin X|\!\})$$
$$s \quad X \;\widehat{=}\; \{t \mapsto s(t) \quad X \mid t \in Is\} \qquad s \setminus X \;\widehat{=}\; \{t \mapsto s(t) \setminus X \mid t \in Is\}$$
$$\Omega \quad X \;\widehat{=}\; \{(t, \chi \quad X) \mid (t, \chi) \in \Omega\} \qquad \Omega \setminus X \;\widehat{=}\; \{(t, \chi \setminus X) \mid (t, \chi) \in \Omega\}$$

We will define a hiding operator on offer relations in section 3.4.11.

The alphabet function Σ returns the set of (untimed) events from a trace or offer relation, or the event component from an offer:

$$\Sigma s \;\widehat{=}\; \{a \mid \exists t \quad a \in s(t)\}$$
$$\Sigma \sqsubseteq \;\widehat{=}\; \{a \mid \exists (t, \chi) \in \text{items} \sqsubseteq \quad a \in \chi\}$$
$$\Sigma(t, \chi) \;\widehat{=}\; \chi$$

The operators $+$ and $-$ temporally shift their arguments forwards or backwards through time:

$$s + t \;\widehat{=}\; \{t + t' \mapsto s(t') \mid t' \in Is\}$$
$$s - t \;\widehat{=}\; \{t' - t \mapsto s(t') \mid t' \in Is \wedge t' \quad t\}$$
$$\sqsubseteq + t \;\widehat{=}\; \sqsubseteq' \quad \text{where } (t + t', \chi) \sqsubseteq' (t + t', \psi) \Leftrightarrow (t', \chi) \sqsubseteq (t', \psi)$$
$$\sqsubseteq - t \;\widehat{=}\; \sqsubseteq' \quad \text{where } (t' - t, \chi) \sqsubseteq' (t' - t, \psi) \Leftrightarrow (t', \chi) \sqsubseteq (t', \psi) \wedge t' \quad t$$
$$\Omega + t \;\widehat{=}\; \{(t' + t, \chi) \mid (t', \chi) \in \Omega\}$$
$$\Omega - t \;\widehat{=}\; \{(t' - t, \chi) \mid (t', \chi) \in \Omega \wedge t' \quad t\}$$

Recall the definition of the function items which returns the set of all offers of an offer relation:

$$\text{items} \sqsubseteq \;\widehat{=}\; \{w \mid \exists v \quad w \sqsubseteq v \vee v \sqsubseteq w\}$$

It is useful to define an operator $\otimes : \quad (TIME) \times \text{seq}(\text{bag}\,\Sigma) \rightarrow OFFREL$ which we will use for representing offer relations: $I \otimes \langle \chi_0, \chi_1, \ldots, \chi_{n-1} \rangle$ represents the offer relation $\sqsubseteq$, such that for all times t during I, $(t, \chi_0) \sqsupset (t, \chi_1) \sqsupset (t, \chi_2) \sqsupset \ldots \sqsupset (t, \chi_{n-1})$.

$$\forall I : \quad (TIME) ; d : \text{seq}(\text{bag}\,\Sigma) \quad I \otimes d = \sqsubseteq$$
$$\text{where } (t, \chi) \sqsubseteq (t', \psi) \Leftrightarrow t = t' \in I \wedge \exists i, j \quad 0 \quad i \quad j < \#d \wedge d(i) = \psi \wedge d(j) = \chi$$

We denote the maximum elements of Ω under $\sqsubseteq$ by $\sqcup_{\sqsubseteq}\Omega$:

$$\sqcup_{\sqsubseteq}\Omega \mathrel{\widehat=} \{(t,\chi) \in \Omega \cap \mathrm{items}\sqsubseteq \mid \forall \psi \quad (t,\psi) \in \mathrm{items}\sqsubseteq \cap\, \Omega \Rightarrow (t,\psi) \sqsubseteq (t,\chi)\}$$

Note that $\sqcup_{\sqsubseteq}\Omega$ is a set of offers, one offer for each time during the duration of Ω, and so can be thought of as a trace — namely the trace where at each instant the element of Ω that is maximal under $\sqsubseteq$ is performed. This will be the trace that a process with offer relation $\sqsubseteq$ will perform when placed in an environment Ω.

3.3.3 Possible behaviours

Only certain behaviours $(\tau, \sqsubseteq, s)$ are possible. We want to limit our attention to those that satisfy a number of healthiness conditions which express some of our intuitions abont how a process should behave. Proving that our semantic definitions do satisfy these conditions will improve our confidence in the model. On the other hand, failure to prove the conditions suggests that something is wrong: when we were first developing the semantic theory, we experimented with several plausible-looking models, only to find that in these models there seemed to be no way of defining the constructs of the language in such a way that the healthiness conditions were satisfied; thus these models had to be abandoned or refined, until we eventually hit upon what we believe to be the correct one.

We define the space *BEH* of possihle behaviours to he those triples $(\tau, \sqsubseteq, s)$ of type $TIME \times OFFREL \times TT$ satisfying the following healthiness conditions:

A1. $I\sqsubseteq = Is = [0, \tau]$

A2. $\forall t \quad \tau \quad s \uparrow t \in \mathrm{items}\sqsubseteq$

A3. $(t_0 < t_1 \wedge \forall t \in (t_0, t_1) \quad (t,\chi) \in \mathrm{items}\sqsubseteq) \Rightarrow (t_0,\chi) \uplus s \uparrow t_0 \sqsupseteq s \uparrow t_0$

A4. $(t_0 < t_1 \wedge \forall t \in (t_0, t_1) \quad (t,\chi) \in \mathrm{items}\sqsubseteq) \Rightarrow (t_1,\chi) \in \mathrm{items}\sqsubseteq$

A5. $(t,\chi) \in \mathrm{items}\sqsubseteq \wedge\, \psi \subseteq \chi \Rightarrow (t,\psi) \in \mathrm{items}\sqsubseteq$

A6. $v \uplus w \uplus w' \sqsupseteq v' \wedge (v \uplus w \uplus w') \cap v' = v \cap v' \Rightarrow (v \uplus w \sqsupseteq v' \vee v \uplus w' \sqsupseteq v')$

A7. $\# s < \infty$

A8. $\exists k : \quad ; I_0, \ldots, I_{k-1} \in TINT$
$I_0, \ldots I_{k-1}$ partition $[0, \tau]$
$\wedge \forall i : 0 \,..\, k-1 \,;\, t, t' \in I_i \,;\, \chi, \psi \in \mathrm{bag}\,\Sigma \quad (t,\chi) \sqsubseteq (t,\psi) \Leftrightarrow (t',\chi) \sqsubseteq (t',\psi)$

We discuss the eight healthiness conditions in turn:

A1. If a process is observed up until time τ, then the time intervals of the trace and offer relation must be the interval $[0, \tau]$.

A2. $\mathrm{items}\sqsubseteq$ is the set of offers that the process is willing to perform: a process can only perform offers from this set.

A3. If a bag χ is offered throughout some open interval beginning at t_0, then at t_0 this bag is offered along with whatever was performed at that time $((t_0, \chi) \uplus s \uparrow t_0)$. Further, the process would have rather performed $(t_0, \chi) \uplus s \uparrow t_0$ to what it did perform.

The condition says something about the time interval over which a process is willing to perform a particular action: namely that this interval is closed on the left. In other words there is a particular time at which an action is made available.

This condition is necessary to avoid processes such as the one that offers a bag χ during $(0, 1)$; if the environment offers χ from time 0 onwards, then it is unclear when it should be performed. The axiom says that if χ is offered throughout $(0, 1)$ then it is offered at 0 along with what was performed then.

To understand why $(t_0, \chi) \uplus s \uparrow t_0$ is offered *stronger* than $s \uparrow t_0$ consider the following situation. Suppose the process P performs a b at time 0 then offers an a during $(0, 1)$, but offers the bag $\{\!|a, b|\!\}$ *weaker* than $\{\!|b|\!\}$ at 0. Suppose this process is in an environment Ω that is willing to perform $\{\!|a, b|\!\}$, $\{\!|b|\!\}$ or $\{\!|a|\!\}$ at time 0, and $\{\!|a|\!\}$ at any time after 0. Then P will perform a b at 0, but there will then be no sensible choice as to when the a can be performed. This axiom (along with a similar condition on environmental offers, presented in section 3.3.4) prevents situations like this from arising: if a process offers an a during $(0, 1)$ after performing a b at time 0, then it should have offered $\{\!|a, b|\!\}$ *stronger* than $\{\!|b|\!\}$ at 0; in environment Ω it would have performed $\{\!|a, b|\!\}$ at time 0.

In chapter 6 we will consider the timed failures that are related to a particular behaviour in the Prioritized Model. This condition will be used to show that the refusal set of a process is open on the right.

A4. If a process offers a bag χ at all times just before t_1, then it also offers χ at t_1. The condition also says something about the time interval over which a process is willing to perform a particular action: namely that the interval is closed on the right. In other words, the offer is still available at the moment at which it is withdrawn.

A5. The offers of a process are subbag closed: if it is willing to perform some bag χ then it is willing to perform any subbag of χ.

A6. To understand this condition it is useful to consider what it means in the prioritized model for a bag of events to be refused. In the classical models of CSP, events are refused if the process can not perform them *in addition* to what it does perform. The obvious adaptation of this to the prioritized model is that a process refuses a bag of events χ at time t if it prefers not to perform χ *in addition* to what it does perform, that is:

$$s \uparrow t \uplus (t, \chi) \not\sqsupseteq s \uparrow t$$

The condition A6 implies the following:

$$v \uplus w \not\sqsupseteq v \wedge v \uplus w' \not\sqsupseteq v \Rightarrow v \uplus w \uplus w' \not\sqsupseteq v \qquad (*)$$

which says that if the process can refuse w while performing v, and can refuse w' while performing v, then it can refuse w and w' when they are offered together.

However, it turns out that this condition is not quite strong enough to prove directly by structural induction. Consider an offer relation with

$$\{\!|a, b|\!\} \sqsupset \{\!|c|\!\} \sqsupset \{\!|a|\!\} \sqsupset \{\!|b|\!\} \sqsupset \{\!|\,|\!\}$$

This relation satisfies $(*)$, but not A6. If we were to hide c from the above offer relation we would get an offer relation with

$$\{\!|a, b|\!\} \sqsupset \{\!|\,|\!\} \sqsupset \{\!|a|\!\} \sqsupset \{\!|b|\!\}$$

which fails the condition $(*)$. We therefore take the stronger condition A6, and deduce $(*)$ as a consequence.

The condition $(v \uplus w \uplus w') \cap v' = v \cap v'$ in the statement of A6 says that v is a subset of $v \uplus w \uplus w'$ that contains as many members of v' as possible (and possibly events from outside v' as well). It is worth noting that this condition is always satisfied if $v' \subseteq v$.

A7. The process can only perform a finite number of events in a finite time. Later we will show that for each process, there is a finite bouud on the number of events that can be performed by a given time.

A8. The offer relation changes shape a finite number of times: there is a finite number of time intervals $I_0, \ldots, I_{k-1}$ such that the offer relation does not change shape during each interval I_i. Later we will show that for each process, there is a finite bound on the number of times that the offer relation can change shape within a given time.

Using these conditions, we can show that the empty bag is always offered:

Theorem 3.3.4: $\forall (\tau, \sqsubseteq, s) \in BEH \quad \forall t \in [0, \tau] \quad (t, \{\!|\,|\!\}) \in \text{items} \sqsubseteq.$ ♡

Proof: By condition A1, $\text{dom}\, s = [0, \tau]$, so for all $t \in [0, \tau]$, $s \uparrow t \in \text{items} \sqsubseteq$ by condition A2. Now $\{\!|\,|\!\} \subseteq \Sigma(s \uparrow t)$, so by condition A5 we have $(t, \{\!|\,|\!\}) \in \text{items} \sqsubseteq$. □

In some circumstances a process can offer a bag of events *weaker* than the empty bag; the following theorem says that a process can only do this at isolated times rather than throughout some interval. This is related to our assumption about maximal progress: as we will see later, offering an empty bag stronger than a non-empty bag corresponds to hidden events being available; these hidden events must either be performed or withdrawn immediately, which means that the empty bag cannot continue to be offered stronger than a non-empty bag throughout some interval.

Theorem 3.3.5: $\quad t_0, t_1, \chi \quad t_0 < t_1 \wedge \forall t \in (t_0, t_1) \quad (t, \chi) \sqsubset (t, \{\!|\,|\!\})$ ♡

Proof: Suppose for a contradiction that there is some t_0, t_1 and χ such that $t_0 < t_1$ and

$$\forall t \in (t_0, t_1) \quad (t, \chi) \sqsubset (t, \{\!|\,|\!\}) \qquad (*)$$

Pick $t_0' \in (t_0, t_1)$ such that $s \uparrow t_0' = (t_0', \{\!|\,|\!\})$: such a t_0' exists by condition A7. Then $\forall t \in (t_0', t_1) \quad (t, \chi) \in \text{items} \sqsubseteq$, so by condition A3 we have $s \uparrow t_0' \uplus (t_0', \chi) \sqsupseteq s \uparrow t_0'$. But this contradicts $(*)$ since $s \uparrow t_0' = (t_0', \{\!|\,|\!\})$. □

3.3.4 Environmental offers

The behaviour of a process is obviously dependent upon the environment in which it executes. In this section we discuss how we model tbe environment. Our representation of the environment will become particularly important in the next chapter when we extend our semantic model to include probabilistic behaviour.

We will represent an environment for a process P by a set of offers: the set of offers that the environment will allow. This set will depend upon the other components of the system, how P is combined with the other components, and the environment for the whole system.

As an example, consider the process $P \mathrel{\widehat{=}} (a \Box b) \setminus a$ in parallel with $Q \mathrel{\widehat{=}} b \longrightarrow STOP$. with Q the master, in an environment that allows a b at time 0. It should be obvious that this b will be performed. But in what environment does P execute? It cannot be in an environment that allows idling, for if it were then it would have performed the a silently. We are forced to conclude that P is in an environment that allows a b, but does not allow idling at time 0.

We shall say that a behaviour $(\tau, \sqsubseteq, s)$ is compatible with an environmental offer Ω (of type (OFF)) if $(\tau, \sqsubseteq, s)$ could have resulted from the environment offering Ω.

Definition 3.3.6: The behaviour $(\tau, \sqsubseteq, s)$ is compatible with the enviroumental offer Ω, written $(\tau, \sqsubseteq, s)$ *compat* Ω, if:

1. $I\Omega = [0, \tau]$
2. $\forall t \quad s \uparrow t = \sqcup_{\sqsubseteq} \Omega \uparrow t$
3. $\exists I_0, \ldots, I_{k-1} \in TINT, X_0, \ldots, X_{k-1} \in \; (\text{bag}\,\Sigma) \quad \Omega = \bigcup\{I_i \times X_i \mid i \in 0 \,..\, k-1\}$
4. $(\forall t \in (t_0, t_1) \quad (t, \chi) \in \Omega) \Rightarrow (t_0, \chi) \uplus s \uparrow t_0 \in \Omega$

where $TINT$ is the set off all time intervals (open, closed or half open). $\diamond$

These conditions state that:

1. The duration of the environmental offer is the same as the duration of the behaviour;
2. At all times, the process performs the element of the environmental offer that is maximal under its offer relation: in other words, the process picks the offer that it prefers;
3. The set of offers changes only finitely often. Note that this condition is independent of the behaviour $(\tau, \sqsubseteq, s)$ — we include it here for the sake of convenience;
4. If a bag of events χ is offered throughout some open time interval beginning at t_0, then at t_0 the environment must have offered χ along with the events of s. In other words, the duration of an offer is closed on the left: offers become available at a particular instant. This condition is necessary to avoid an environment such as the one that offers an a during the period $(0, 1]$; if a process that is willing to perform an a from time 0 onwards is placed in this environment, then there is no sensible choice as to when the event should be performed.

Of these, condition 2 is perhaps the most important. It describes the way that a process chooses the events it performs. At each instant the environment is willing to perform any one of a number of bags of events; the process takes its pick from these by choosing the bag that is strongest under its offer relation.

It should be noted that there is no ordering on the environmental offer: it is simply a set of offers from which a process is able to make its choice.

In general, we will allow the environmental offer to be a function of the observed behaviour. This fits in with our intuition of the environment for process P being dependent upon the other processes in the system: different behaviours of P will cause the other components to act in different ways, and so will cause different environmental offers in the future. In general, it is enough to allow the environment to depend upon the offer relation of the process. When we want to stress that environment Ω is a function of the offer relation $\sqsubseteq$, we will write $\Omega(\sqsubseteq)$. We shall insist that an environment cannot depend upon the future behaviour of the process (i.e. it is not clairvoyant):

$$\sqsubseteq \quad t = \sqsubseteq' \quad t \Rightarrow \Omega(\sqsubseteq) \quad t = \Omega(\sqsubseteq') \quad t$$

We will only be interested in environmental offers that allow the process to act in some manner, even if it only allows the process to idle. For example, we do not want the process $a \longrightarrow STOP$ to operate in an environment that allows neither an a nor idling. We will call the situation where the environment does not allow the process to progress at all a *time deadlock*. We shall call an environment *friendly* if it does not allow time deadlock; this can be formalized as

$$\forall \sqsubseteq \quad \forall t \quad end \sqsubseteq \quad \exists \chi \quad (t, \chi) \in \text{items} \sqsubseteq \cap \Omega(\sqsubseteq)$$

Whatever offer relation the process has, there is some behaviour with this offer relation that is compatible with the environment.

It turns out that, subject to a very reasonable assumption, every process executes in a friendly environment. We thing of a system as being built out of several components. We assume that the system as a whole is in a friendly environment — this fits with our intuition of a system being in an environment provided by an observer who is willing to observe anything. In producing the semantic definitions for the operators we will ensure that if a composite process is in a friendly environment then the subcomponents are also in friendly environments (this was formally proved in [Low91b]). Hence by induction on the structure of the system, we can deduce that every component is in a friendly environment.

We will therefore consider only friendly environments. This is equivalent to taking the following definition for the space of environmental offers:

Definition 3.3.7:

$$EOFF \mathrel{\hat{=}} \{\Omega : OFFREL \rightarrow \quad (OFF) \mid \forall \sqsubseteq \quad \exists \tau . s \quad (\tau, \sqsubseteq, s) \; compat \; \Omega(\sqsubseteq)\}$$

$\diamond$

There is always some behaviour that is compatible with the environmental offer.

3.3.5 The semantic space $\mathcal{M}_{TB}$

We are now ready to define our semantic space. Firstly, we give a name to the space of sets of prioritized behaviours

$$\mathcal{S}_{TB} \mathrel{\widehat{=}} \quad (BEH)$$

$\mathcal{S}_{TB}$ is the space of sets of timed biased behaviours. We define the space $\mathcal{M}_{TB}$ (the Model using Timed, Biased behaviours) to be those sets A of type $\mathcal{S}_{TB}$ satisfying a number of axioms. Intuitively, the set A represents a process that can behave like any of the elements of A. The set A must obey the following axioms:

B1. $\forall \tau \quad 0 \;\; \exists\, n(\tau) \;\; (\tau, \sqsubseteq, s) \in A \Rightarrow \# s \quad n(\tau)$

B2. $\forall \tau \quad 0 \;\; \exists\, n(\tau) \;\; (\tau, \sqsubseteq, s) \in A \Rightarrow \exists\, k \quad n(\tau)\,;\, I_0, \ldots, I_{k-1} \in TINT$
$I_0, \ldots I_{k-1}$ partition $[0, \tau]$
$\wedge \forall\, i : 0 \ldots k-1\,;\, t, t' \in I_i\,;\, \chi, \psi \in \text{bag}\,\Sigma \quad (t, \chi) \sqsubseteq (t, \psi) \Leftrightarrow (t', \chi) \sqsubseteq (t', \psi)$

B3. $(\tau, \sqsubseteq, s) \in A \wedge (t, \chi) \in \text{items} \sqsubseteq \Rightarrow (t, \sqsubseteq \quad t, s \quad t \quad (t, \chi)) \in A$

B4. $\exists \sqsubseteq \quad (0, \sqsubseteq, \prec\succ) \in A$

B5. $\forall (\tau, \sqsubseteq, s) \in A\,;\, \tau' > \tau\,;\, \Omega : EOFF \quad I\Omega = (\tau, \tau'] \Rightarrow$
$\exists \sqsubseteq' \quad \sqsubseteq' \quad \tau = \sqsubseteq \wedge (\tau', \sqsubseteq', s \quad \sqcup_{\sqsubseteq' \not\models \tau}\, \Omega(\sqsubseteq)) \in A$

We discuss each of these axioms in turn:

B1. The number of events that a process can perform in a finite time is uniformly bounded.

B2. The number of times at which an offer relation can change in a finite time is uniformly bounded.

B3. A process is able to perform any bag of events that it offers.

B4. The process can perform the empty trace up until time 0.

B5. Any behaviour can be extended in time: if the process can perform some behaviour $(\tau, \sqsubseteq, s)$ up until time τ, then if it is observed until τ', it can have some offer relation $\sqsubseteq'$, which agrees with $\sqsubseteq$ until τ, and at each instant after τ will perform the element of the environmental offer that it prefers.

3.3.6 Laws

The following law can be deduced from the axioms. If a process can have a particular behaviour, then it can perform any prefix of that behaviour:

Theorem 3.3.8:

$$(\tau, \sqsubseteq, s) \in A \wedge \tau' \quad \tau \Rightarrow (\tau', \sqsubseteq \quad \tau', s \quad \tau') \in A \wedge (\tau', \sqsubseteq \quad \tau', s \quad \tau' \quad (\tau', \{\!\!\}\!\!\})) \in A$$

♡

Proof: Suppose $(\tau, \sqsubseteq, s) \in A \wedge \tau' \quad \tau$. Then by condition A2 on behaviours, $s \uparrow \tau' \in$ items $\sqsubseteq$, and by theorem 3.3.4 $(\tau', \{\!|\,|\!\}) \in$ items $\sqsubseteq$, so by axiom B3 we have

$$(\tau', \sqsubseteq \quad \tau', s \quad \tau' \quad s \uparrow \tau') \in A \qquad \text{and} \qquad (\tau', \sqsubseteq \quad \tau', s \quad \tau' \quad (\tau', \{\!|\,|\!\})) \in A$$

as required. □

3.3.7 Semantic functions

In order to give a semantics to variables we define a space ENV of environments or variable bindings, which contains all functions from VAR, the set of variables, to sets of behaviours:

$$ENV \mathrel{\widehat{=}} VAR \to S_{TB}$$

We will write $\rho\ X$ for the value assigned to variable X in environment ρ.

We shall define a function $\mathcal{A}_{BT} : BTCSP \to ENV \to S_{TB}$ such that $\mathcal{A}_{BT}\ P\ \rho$ gives the set of possible behaviours of process P given variable binding ρ. We can give semantics to syntactic substitution as follows:

$$\mathcal{A}_{BT}\ P[Q/X]\ \rho = \mathcal{A}_{BT}\ P\ \rho[\mathcal{A}_{BT}\ Q\ \rho/X]$$

where $\rho[Y/X]$ is the environment obtained from ρ by setting X to Y:

$$\rho[Y/X]\ Z \mathrel{\widehat{=}} \begin{cases} Y & \text{if } Z = X \\ \rho\ Z & \text{otherwise} \end{cases}$$

A BTCSP process is a BTCSP term with no free variables. Its semantics will be independent of the environment, and so in this case it is sensible to omit reference to the environment.

In the following section we give definitions for $\mathcal{A}_{BT}$ for each of the operators; in most cases the crux of the definition will be the explanation of how the offer relation of a composite process results from the offer relations of its subcomponents. The definitions were proved sound (i.e. they respect the axioms) in [Low91b].

We will state a number of laws that can be shown to hold of our processes, and also show which laws do not hold. Most of the laws were proved sound in [Low91a].

3.4 Semantic definitions

Throughout this section we will take $A_P \mathrel{\widehat{=}} \mathcal{A}_{BT}\ P\ \rho$, $A_Q \mathrel{\widehat{=}} \mathcal{A}_{BT}\ Q\ \rho$.

3.4.1 *STOP*

The process $STOP$ always performs the empty trace and offers only the empty bag of events:

$$\mathcal{A}_{BT}\ STOP\ \rho \mathrel{\widehat{=}} \{(\tau, [0,\tau] \circledS \langle\{\!|\,|\!\}\rangle, \prec\succ) \mid \tau \in TIME\}$$

3.4.2 *WAIT t*

The process *WAIT t* behaves as follows:

- for observations ending before t, nothing is performed and only the empty bag of events is offered;
- if the environment does not offer at or after t then it performs the empty trace and offers from t onwards;
- if is offered by the environment at or after t then it is immediately performed; will be offered from time t until it is performed.

This gives the following definition:

$$\begin{aligned}\mathcal{A}_{BT}\ \ WAIT\ t\ \ \rho \mathrel{\hat=} & \{(\tau, [0,\tau] \otimes \langle\{\!\!\}\!\!\}\rangle, \prec\succ) \mid \tau < t\} \\ & \cup\{(\tau, [0,t) \otimes \langle\{\!\!\}\!\!\}\rangle \ \ [t,\tau] \otimes \langle\{\!\!\} \ \ \!\!\}, \{\!\!\}\!\!\}\rangle, \prec\succ) \mid \tau \ \ t\} \\ & \cup\{(\tau, [0,t) \otimes \langle\{\!\!\}\!\!\}\rangle \ \ [t,t'] \otimes \langle\{\!\!\} \ \ \!\!\}, \{\!\!\}\!\!\}\rangle \ \ (t',\tau] \otimes \langle\{\!\!\}\!\!\}\rangle, \prec(t', \ \ )\succ) \mid \\ & \qquad t \ \ t' \ \ \tau\}\end{aligned}$$

3.4.3 *SKIP*

SKIP is equivalent to *WAIT 0*, so we have the following definition:

$$\begin{aligned}\mathcal{A}_{BT}\ \ SKIP\ \ \rho \mathrel{\hat=} & \{(\tau, [0,\tau] \otimes \langle\{\!\!\} \ \ \!\!\}, \{\!\!\}\!\!\}\rangle, \prec\succ)\} \\ & \cup\{(\tau, [0,t] \otimes \langle\{\!\!\} \ \ \!\!\}, \{\!\!\}\!\!\}\rangle \ \ (t,\tau] \otimes \langle\{\!\!\}\!\!\}\rangle, \prec(t, \ \ )\succ) \mid t \ \ \tau\}\end{aligned}$$

3.4.4 Variables

We give semantics to the clause X in the obvious way:

$$\mathcal{A}_{BT}\ \ X\ \ \rho \mathrel{\hat=} \rho\ X$$

3.4.5 Prefixing

The process $a \xrightarrow{0} P$ should offer an a until it is performed, and then act like P. In order for this to fit with our intuition of causality, we insist that P is unable to perform any events at time 0.

$$\begin{aligned}\mathcal{A}_{BT}\ \ a \xrightarrow{0} P\ \ \rho \mathrel{\hat=} & \{(\tau, [0,\tau] \otimes \langle\{\!\!\}a\}\!\!\}, \{\!\!\}\!\!\}\rangle, \prec\succ)\} \\ & \cup\{(\tau, [0,t] \otimes \langle\{\!\!\}a\}\!\!\}, \{\!\!\}\!\!\}\rangle \ \ \sqsubseteq_P + t, (t,a) \ \ s_P + t) \mid \\ & \qquad (\tau - t, 0 \otimes \langle\{\!\!\}\!\!\}\rangle \ \ \sqsubseteq_P, \prec\succ \ \ s_P) \in A_P \wedge \tau \ \ t\}\end{aligned}$$

We define the general prefix operator by $a \xrightarrow{t} P \mathrel{\hat=} a \xrightarrow{0} WAIT\ t\ ; P$.

3.4.6 External choice

Consider the process $P \boxdot Q$. We want to derive a definition for the offer relation of $P \boxdot Q$ in terms of the offer relations of P and Q. We begin by considering an example. Suppose P has offer relation $\sqsubseteq_P$ and Q has offer relation $\sqsubseteq_Q$, with $\{\!|a|\!\} \sqsupset_P \{\!||\!\} \sqsupset_P \{\!|b|\!\}$ and $\{\!|c|\!\} \sqsupset_Q \{\!|a|\!\} \sqsupset_Q \{\!||\!\} \sqsupset_Q \{\!|d|\!\}$. Then:

- If the environment offers $\{\!|a|\!\}$ then P will perform it;
- If the environment does not offer $\{\!|a|\!\}$, then P may idle and Q may perform $\{\!|c|\!\}$, $\{\!||\!\}$ or $\{\!|d|\!\}$. Note that even if the environment doesn't allow idling at some time t — for example if it offers only $\{\!|c|\!\}$ or $\{\!|d|\!\}$ — then P may idle at time t while Q performs $\{\!|c|\!\}$ or $\{\!|d|\!\}$. Note also that Q cannot perform $\{\!|a|\!\}$ since if the environment offers $\{\!|a|\!\}$ then it would be performed by P.
- If none of these are possible, then P will perform $\{\!|b|\!\}$.

Hence $P \boxdot Q$ has an offer relation with $\{\!|a|\!\} \sqsupset \{\!|c|\!\} \sqsupset \{\!||\!\} \sqsupset \{\!|d|\!\} \sqsupset \{\!|b|\!\}$.
In general, the offer relation of $P \boxdot Q$ is formed by

1. taking P's offer relation ($\{\!|a|\!\} \sqsupset \{\!||\!\} \sqsupset \{\!|b|\!\}$ in our example);
2. replacing the occurrence of $\{\!||\!\}$ with Q's offer relation (to get $\{\!|a|\!\} \sqsupset \{\!|c|\!\} \sqsupset \{\!|a|\!\} \sqsupset \{\!||\!\} \sqsupset \{\!|d|\!\} \sqsupset \{\!|b|\!\}$ in our example);
3. for each bag that occurs twice, removing the lower copy (to get $\{\!|a|\!\} \sqsupset \{\!|c|\!\} \sqsupset \{\!||\!\} \sqsupset \{\!|d|\!\} \sqsupset \{\!|b|\!\}$).

In general, $P \boxdot Q$ will perform the offer w if the environment offers w and

- P would rather perform w than idle and the environment offers nothing that P prefers to w;
- P chooses to idle, Q offers w and the environment offers nothing that Q prefers to w; *or*
- Q doesn't offer w, P would rather idle than perform w but the environment does not allow idling and does not offer anything that Q could perform nor anything that P prefers to w.

The process should offer w more strongly than v if

- P prefers w to idling and v is either offered by Q but not P, or offered by P less strongly than w;
- P prefers neither v nor w to idling, Q offers w and either
  - Q prefers w to v; *or*
  - v is offered by P but not Q;

or

- P offers v weaker than w, but would rather idle, and Q offers neither v or w.

Hence if P has offer relation $\sqsubseteq_P$ and Q has offer relation $\sqsubseteq_Q$ then $P \sqcap\!\!\!\sqcup Q$ has offer relation $\sqsubseteq_P \sqcap\!\!\!\sqcup \sqsubseteq_Q$, where the operator $\sqcap\!\!\!\sqcup : OFFREL \times OFFREL \to OFFREL$ is defined by

Definition 3.4.1 (left-biased choice composition of offer relations) For all offers v and w, if $t = Iv = Iw$ then

$$\begin{aligned} v(\sqsubseteq_P \sqcap\!\!\!\sqcup \sqsubseteq_Q)w \Leftrightarrow\ & w \sqsupset_P (t, \{\!|\,|\!\}) \wedge (v \sqsubseteq_P w \vee v \in \text{items} \sqsubseteq_Q \setminus \text{items} \sqsubseteq_P) \\ \vee\ & v, w \not\sqsupset_P (t, \{\!|\,|\!\}) \wedge w \in \text{items} \sqsubseteq_Q \wedge (v \sqsubseteq_Q w \vee v \in \text{items} \sqsubseteq_P \setminus \text{items} \sqsubseteq_Q) \\ \vee\ & v \sqsubseteq_P w \sqsubset_P (t, \{\!|\,|\!\}) \wedge v, w \notin \text{items} \sqsubseteq_Q \end{aligned}$$

$\diamond$

Note that $\text{items}(\sqsubseteq_P \sqcap\!\!\!\sqcup \sqsubseteq_Q) = \text{items} \sqsubseteq_P \cup \text{items} \sqsubseteq_Q$. Note also that the operator is still defined when the durations of $\sqsubseteq_P$ and $\sqsubseteq_Q$ are different: for example if the relation $\sqsubseteq_P$ is empty after time t, but $\sqsubseteq_Q$ extends beyond this time, then after time t the offer relation $\sqsubseteq_P \sqcap\!\!\!\sqcup \sqsubseteq_Q$ is just the same as $\sqsubseteq_Q$.

Having explained how the offer relation of $P \sqcap\!\!\!\sqcup Q$ is derived from the offer relations of P and Q, we can now derive the semantic definition of the process. The process $P \sqcap\!\!\!\sqcup Q$ can

- perform the empty trace if both P and Q can;
- perform a non empty trace s if P can perform s and Q can perform the empty trace up until time $t = \mathit{begin}\, s$; if the bag χ performed at time t is below the empty bag in P's offer relation, then Q must also be able to reject it (or else Q would have performed χ); i.e. $s \uparrow t \sqsupset_P (t, \{\!|\,|\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_Q$; *or*
- perform a non empty trace s if Q can perform it and P can perform the empty trace up until time $\mathit{begin}\, s$ and P prefers idling to the initial events of s, i.e. $s \uparrow t \not\sqsupset_P (t, \{\!|\,|\!\})$.

This gives the following definition:

$$\begin{aligned} & \mathcal{A}_{BT}\ P \sqcap\!\!\!\sqcup Q\ \rho \mathrel{\hat=} \\ & \quad \{(\tau, \sqsubseteq_P \sqcap\!\!\!\sqcup \sqsubseteq_Q, \prec\succ) \mid (\tau, \sqsubseteq_P, \prec\succ) \in A_P \wedge (\tau, \sqsubseteq_Q, \prec\succ) \in A_Q\} \\ & \quad \cup \{(\tau, \sqsubseteq_P \sqcap\!\!\!\sqcup \sqsubseteq_Q, s) \mid s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge (\tau, \sqsubseteq_P, s) \in A_P \\ & \qquad\qquad \wedge (t, \sqsubseteq_Q, \prec\succ) \in A_Q \wedge (s \uparrow t \sqsupset_P (t, \{\!|\,|\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_Q)\} \\ & \quad \cup \{(\tau, \sqsubseteq_P \sqcap\!\!\!\sqcup \sqsubseteq_Q, s) \mid s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge (t, \sqsubseteq_P, \prec\succ) \in A_P \\ & \qquad\qquad \wedge (\tau, \sqsubseteq_Q, s) \in A_Q \wedge s \uparrow t \not\sqsupset_P (t, \{\!|\,|\!\})\} \end{aligned}$$

We define $P \sqcup\!\!\!\sqcap Q$ by $P \sqcup\!\!\!\sqcap Q \mathrel{\hat=} Q \sqcap\!\!\!\sqcup P$.

We have a number of laws for the choice operators:

Law 3.4.2 (Associativity of external choice)

$$(P \sqcap\!\!\!\sqcup Q) \sqcap\!\!\!\sqcup R = P \sqcap\!\!\!\sqcup (Q \sqcap\!\!\!\sqcup R) \quad \text{and} \quad (P \sqcup\!\!\!\sqcap Q) \sqcup\!\!\!\sqcap R = P \sqcup\!\!\!\sqcap (Q \sqcup\!\!\!\sqcap R)$$

$\triangle$

Law 3.4.3 (*STOP* is an identity of external choice)

$$P \overleftarrow{\Box}\, STOP = P \quad \text{and} \quad STOP \overleftarrow{\Box}\, P = P$$

$\triangle$

Note 3.4.4: The following laws do not hold:

$$P \overleftarrow{\Box} (Q \overrightarrow{\Box} R) = (P \overleftarrow{\Box} Q) \overrightarrow{\Box} R\,; \qquad P \overrightarrow{\Box} (Q \overleftarrow{\Box} R) = (P \overrightarrow{\Box} Q) \overleftarrow{\Box} R$$

Let $P \mathrel{\hat=} a \longrightarrow STOP$, $Q \mathrel{\hat=} b \longrightarrow STOP$, $R \mathrel{\hat=} c \longrightarrow STOP$. Then $P \overleftarrow{\Box} (Q \overrightarrow{\Box} R)$ and $(P \overrightarrow{\Box} Q) \overleftarrow{\Box} R$ will perform an a in preference to a c, whereas $(P \overleftarrow{\Box} Q) \overrightarrow{\Box} R$ and $P \overrightarrow{\Box} (Q \overleftarrow{\Box} R)$ will perform a c in preference to an a. $\diamond$

Note 3.4.5: The external choice operator is not idempotent in this model. Let $P \mathrel{\hat=} a \longrightarrow STOP \sqcap b \longrightarrow STOP$. Then $P \overleftarrow{\Box} P$ can have an offer relation with $(0, \{\!|b|\!\}) \sqsupset (0, \{\!|a|\!\}) \sqsupset (0, \{\!|\,|\!\})$, whereas P cannot have this offer relation. $\diamond$

3.4.7 Parallel composition

We consider now the parallel composition of two processes. We start by considering the left-biased parameterized parallel composition, $P\ {}^X\!\overleftarrow{\|}^Y\ Q$. The offer relation of $P\ {}^X\!\overleftarrow{\|}^Y\ Q$ is derived from the offer relations of P and Q. $P\ {}^X\!\overleftarrow{\|}^Y\ Q$ will offer w if

- P offers $w \quad X$;
- Q offers $w \quad Y$; *and*
- all the events of w are in either X or Y.

w is offered more strongly than v if

- P offers $w \quad X$ more strongly than $v \quad X$; *or*
- $w \quad X = v \quad X$ and Q offers $w \quad Y$ more strongly than $v \quad Y$.

Hence if P has offer relation $\sqsubseteq_P$ and Q has offer relation $\sqsubseteq_Q$ then the offer relation is $\sqsubseteq_P\ {}^X\!\overleftarrow{\|}^Y \sqsubseteq_Q$, where the function ${}^X\!\overleftarrow{\|}^Y : \mathit{OFFREL} \times \mathit{OFFREL} \to \mathit{OFFREL}$ is defined by

Definition 3.4.6 (Left-biased parallel composition of offer relations) For all offers v and w,

$$\begin{aligned}
& v(\sqsubseteq_P\ {}^X\!\overleftarrow{\|}^Y \sqsubseteq_Q)w \Leftrightarrow \\
& \quad (v \quad X \sqsubset_P w \quad X \vee v \quad X = w \quad X \wedge v \quad Y \sqsubseteq_Q w \quad Y) \\
& \quad \wedge v \quad X, w \quad X \in \text{items} \sqsubseteq_P \wedge v \quad Y, w \quad Y \in \text{items} \sqsubseteq_Q \wedge \Sigma v, \Sigma w \subseteq X \cup Y
\end{aligned}$$

$\diamond$

Note that $\text{items}(\sqsubseteq_P {}^{X}\!\parallel^{Y} \sqsubseteq_Q) = \{w \mid w \;\; X \in \text{items} \sqsubseteq_P \wedge w \;\; Y \in \text{items} \sqsubseteq_Q \wedge \Sigma w \subseteq X \cup Y\}$.

$P \,{}^{X}\!\parallel^{Y} Q$ will perform trace s if

- The alphabet of s is contained in $X \cup Y$;
- P can perform $s \;\; X$; *and*
- Q can perform $s \;\; Y$.

Hence, we have the following definition for parallel composition:

$$\begin{array}{l} \mathcal{A}_{BT} \; P \,{}^{X}\!\parallel^{Y} Q \;\; \rho \mathrel{\widehat{=}} \\ \quad \{(\tau, \sqsubseteq_P {}^{X}\!\parallel^{Y} \sqsubseteq_Q, s) \mid (\tau, \sqsubseteq_P, s \;\; X) \in A_P \wedge (\tau, \sqsubseteq_Q, s \;\; Y) \in A_Q \wedge \Sigma s \subseteq X \cup Y\} \end{array}$$

We use this definition to define the other parallel operators:

$$P \,{}^{X}\!\interleave^{Y} Q \mathrel{\widehat{=}} Q \,{}^{Y}\!\parallel^{X} P \qquad P \parallel Q \mathrel{\widehat{=}} P \,{}^{\Sigma}\!\parallel^{\Sigma} Q \qquad P \interleave Q \mathrel{\widehat{=}} P \,{}^{\Sigma}\!\interleave^{\Sigma} Q$$

This gives the following

$$\mathcal{A}_{BT} \;\; P \parallel Q \;\; \rho = \{(\tau, \sqsubseteq_P \parallel \sqsubseteq_Q, s) \mid (\tau, \sqsubseteq_P, s) \in A_P \wedge (\tau, \sqsubseteq_Q, s) \in A_Q\}$$

where the parallel composition of offers is defined by

$$v(\sqsubseteq_P \parallel \sqsubseteq_Q) w \Leftrightarrow v \sqsubseteq_P w \wedge v, w \in \text{items} \sqsubseteq_Q$$

Note that $\text{items}(\sqsubseteq_P \parallel \sqsubseteq_Q) = \text{items} \sqsubseteq_P \cap \text{items} \sqsubseteq_Q$.

A number of laws hold for the parallel operators:

Law 3.4.7 (Associativity of parallel composition) The following laws hold:

$$\begin{array}{rcl} P \,{}^{X}\!\parallel^{Y\cup Z} (Q \,{}^{Y}\!\parallel^{Z} R) & = & (P \,{}^{X}\!\parallel^{Y} Q) \,{}^{X\cup Y}\!\parallel^{Z} R \\ P \,{}^{X}\!\interleave^{Y\cup Z} (Q \,{}^{Y}\!\interleave^{Z} R) & = & (P \,{}^{X}\!\interleave^{Y} Q) \,{}^{X\cup Y}\!\interleave^{Z} R \\ P \interleave (Q \parallel R) & = & (P \interleave Q) \parallel R \\ P \parallel (Q \parallel R) & = & P \parallel (Q \interleave R) \end{array}$$

$\triangle$

Law 3.4.8 (*STOP* is a zero of parallel composition) $P \parallel STOP = STOP$ and $P \interleave STOP = STOP$. $\triangle$

Law 3.4.9 (Communication) The following laws for communication hold:

$$\begin{array}{rl} (a \longrightarrow P) \parallel (a \longrightarrow Q) = a \longrightarrow (P \parallel Q) & (a \longrightarrow P) \interleave (a \longrightarrow Q) = a \longrightarrow (P \interleave Q) \\ (a \longrightarrow P) \parallel (b \longrightarrow Q) = STOP & (a \longrightarrow P) \interleave (b \longrightarrow Q) = STOP \end{array}$$

$\triangle$

Note 3.4.10: We do not have the following laws:

$$
\begin{aligned}
P\ {}^{X}\overrightarrow{\parallel}{}^{Y\cup Z}\ (Q\ {}^{Y}\overleftarrow{\parallel}{}^{Z}\ R) &= (P\ {}^{X}\overrightarrow{\parallel}{}^{Y}\ Q)\ {}^{X\cup Y}\overleftarrow{\parallel}{}^{Z}\ R \\
P\ {}^{X}\overleftarrow{\parallel}{}^{Y\cup Z}\ (Q\ {}^{Y}\overleftarrow{\parallel}{}^{Z}\ R) &= P\ {}^{X}\overleftarrow{\parallel}{}^{Y\cup Z}\ (Q\ {}^{Y}\overrightarrow{\parallel}{}^{Z}\ R)
\end{aligned}
$$

Let $P \mathrel{\widehat{=}} a \longrightarrow STOP$, $Q \mathrel{\widehat{=}} b \longrightarrow STOP$, $R \mathrel{\widehat{=}} c \longrightarrow STOP$, $X \mathrel{\widehat{=}} \{a\}$, $Y \mathrel{\widehat{=}} \{b\}$, $Z \mathrel{\widehat{=}} \{c\}$. Then:

- $P\ {}^{X}\overrightarrow{\parallel}{}^{Y\cup Z}\ (Q\ {}^{Y}\overleftarrow{\parallel}{}^{Z}\ R)$ prefers a c to an a, whereas $(P\ {}^{X}\overrightarrow{\parallel}{}^{Y}\ Q)\ {}^{X\cup Y}\overleftarrow{\parallel}{}^{Z}\ R$ prefers an a to a c;
- $P\ {}^{X}\overleftarrow{\parallel}{}^{Y\cup Z}\ (Q\ {}^{Y}\overleftarrow{\parallel}{}^{Z}\ R)$ prefers a b to a c, whereas $P\ {}^{X}\overleftarrow{\parallel}{}^{Y\cup Z}\ (Q\ {}^{Y}\overrightarrow{\parallel}{}^{Z}\ R)$ prefers a c to a b.

$\diamond$

Note 3.4.11: We do not have the law $P \overleftarrow{\parallel} (Q \overrightarrow{\parallel} R) = (P \overleftarrow{\parallel} Q) \overrightarrow{\parallel} R$. Let $P \mathrel{\widehat{=}} a \mathbin{\Box\!\!\!\rightarrow} b$, $Q \mathrel{\widehat{=}} a \mathbin{\Box\!\!\!\rightarrow} b$, $R \mathrel{\widehat{=}} a \mathbin{\Box\!\!\!\rightarrow} b$. Then $P \overleftarrow{\parallel} (Q \overrightarrow{\parallel} R)$ prefers an a to a b, whereas $(P \overleftarrow{\parallel} Q) \overrightarrow{\parallel} R$ prefers a b to an a. $\diamond$

3.4.8 Interleaving

We want to derive a definition for the offer relation of $P \longleftarrow Q$ in terms of the offer relations of P and Q. We begin by considering the question

If $P \longleftarrow Q$ offers w, then what do P and Q offer?

It is clear that P must offer some suboffer of w, and Q must offer the rest of w. Let w_P be the suboffer of w that P offers strongest subject to the condition that Q can perform the rest of w. Let w_Q be the rest of w. We make the assumption that $P \longleftarrow Q$ offering w corresponds to P offering w_P and Q offering w_Q: since P is the master, it chooses the suboffer of w that it prefers. We define an operator $\Uparrow_{\sqsubseteq_P,\sqsubseteq_Q}$ which returns the subset of its argument that is offered strongest by $\sqsubseteq_P$ subject to the condition that the rest of the argument is offered by $\sqsubseteq_Q$.

$$
\begin{aligned}
&(\exists\, w_P \in \text{items} \sqsubseteq_P, w_Q \in \text{items} \sqsubseteq_Q \quad w = w_P \uplus w_Q) \Rightarrow \\
&\qquad \Uparrow_{\sqsubseteq_P,\sqsubseteq_Q} w = \sqcup_{\sqsubseteq_P} \{w'_P \in \text{items} \sqsubseteq_P \mid w'_P \subseteq w \wedge w - w'_P \in \text{items} \sqsubseteq_Q\}
\end{aligned}
$$

It will be useful to define an operator that returns the rest of the offer:

$$
(\exists\, w_P \in \text{items} \sqsubseteq_P, w_Q \in \text{items} \sqsubseteq_Q \quad w = w_P \uplus w_Q) \quad \Rightarrow \quad \Downarrow_{\sqsubseteq_P,\sqsubseteq_Q} w = w - \Uparrow_{\sqsubseteq_P,\sqsubseteq_Q} w
$$

Let w_P and w_Q be the suboffers of w performed by P and Q respectively, i.e. $\Uparrow_{\sqsubseteq_P,\sqsubseteq_Q} w$ and $\Downarrow_{\sqsubseteq_P,\sqsubseteq_Q} w$. Let v_P and v_Q be the corresponding suboffers of v. Then $P \longleftarrow Q$ offers w more strongly than v if

- P offers w_P strictly stronger than v_P, or
- $w_P = v_P$ and Q offers w_Q stronger than v_Q.

Hence, if P and Q have offer relations $\sqsubseteq_P$ and $\sqsubseteq_Q$, the offer relation for $P \longleftarrow Q$ is $\sqsubseteq_P \longleftarrow \sqsubseteq_Q$ where $\longleftarrow$ is defined by

Definition 3.4.12 (Interleaving of offer relations) For all offers v and w, if

$$\begin{array}{l} \exists\, v'_P \in \text{items} \sqsubseteq_P, v'_Q \in \text{items} \sqsubseteq_Q \quad v = v'_P \uplus v'_Q \\ \wedge \exists\, w'_P \in \text{items} \sqsubseteq_P, w'_Q \in \text{items} \sqsubseteq_Q \quad w = w'_P \uplus w'_Q \end{array}$$

then

$$v(\sqsubseteq_P \longleftarrow \sqsubseteq_Q)w \Leftrightarrow v_P \sqsubset_P w_P \vee v_P = w_P \wedge v_Q \sqsubseteq_Q w_Q$$

where

$$v_P = \Uparrow_{\sqsubseteq_P,\sqsubseteq_Q} v \quad v_Q = \Downarrow_{\sqsubseteq_P,\sqsubseteq_Q} v \quad w_P = \Uparrow_{\sqsubseteq_P,\sqsubseteq_Q} w \quad w_Q = \Downarrow_{\sqsubseteq_P,\sqsubseteq_Q} w$$

$\diamond$

Note that items$(\sqsubseteq_P \longleftarrow \sqsubseteq_Q) = \{w_P \uplus w_Q \mid w_P \in \text{items} \sqsubseteq_P \wedge w_Q \in \text{items} \sqsubseteq_Q\}$.

$P \longleftarrow Q$ can perform trace s if, at all times t, P can perform some subbag of $s \uparrow t$ and Q can perform the rest of $s \uparrow t$. In particular, P performs that subbag of $s \uparrow t$ that it offers strongest subject to the condition that Q can perform the rest of $s \uparrow t$. We extend the $\Uparrow_{\sqsubseteq_P,\sqsubseteq_Q}$ and $\Downarrow_{\sqsubseteq_P,\sqsubseteq_Q}$ operators to traces:

$$\Uparrow_{\sqsubseteq_P,\sqsubseteq_Q} s \mathrel{\widehat{=}} \{t \mapsto \Uparrow_{\sqsubseteq_P,\sqsubseteq_Q}(s \uparrow t) \mid t \in Is\} \qquad \Downarrow_{\sqsubseteq_P,\sqsubseteq_Q} s \mathrel{\widehat{=}} \{t \mapsto \Downarrow_{\sqsubseteq_P,\sqsubseteq_Q}(s \uparrow t) \mid t \in Is\}$$

We then have the following definition:

$$\mathcal{A}_{BT}\ P \longleftarrow Q\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq_P \longleftarrow \sqsubseteq_Q, s)) \mid (\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P,\sqsubseteq_Q} s) \in A_P \wedge (\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P,\sqsubseteq_Q} s) \in A_Q\}$$

The right biased interleave operator is defined by $P \longrightarrow Q \mathrel{\widehat{=}} Q \longleftarrow P$.

We have a number of laws for interleaving:

Law 3.4.13 (Associativity of interleaving)

$$P \longleftarrow (Q \longleftarrow R) = (P \longleftarrow Q) \longleftarrow R \quad \text{and} \quad P \longrightarrow (Q \longrightarrow R) = (P \longrightarrow Q) \longrightarrow R$$

$\triangle$

Law 3.4.14 (*STOP* is an identity of interleaving)

$$P \longleftarrow STOP = P \quad \text{and} \quad P \longrightarrow STOP = P$$

$\triangle$

Note 3.4.15: We do not have the following laws:

$$P \longleftarrow (Q \longrightarrow R) = (P \longleftarrow Q) \longrightarrow R \qquad P \longrightarrow (Q \longleftarrow R) = (P \longrightarrow Q) \longleftarrow R$$

Let $P \mathrel{\widehat{=}} a \longrightarrow STOP$, $Q \mathrel{\widehat{=}} b \longrightarrow STOP$, $R \mathrel{\widehat{=}} c \longrightarrow STOP$. Then $P \longleftarrow (Q \longrightarrow R)$ and $(P \longrightarrow Q) \longleftarrow R$ prefer a to c whereas $(P \longleftarrow Q) \longrightarrow R$ and $P \longrightarrow (Q \longleftarrow R)$ prefer c to a.
$\diamond$

3.4.9 Communicating parallel

The process $P \underset{C}{\Vert} Q$ executes the processes P and Q in parallel, synchronising on events from C and interleaving on all other events. It can be defined by

$$P \underset{C}{\Vert} Q \mathrel{\widehat{=}} c(l(P)\ {}^{A}\Vert^{B}\ r(Q))$$

where

$$l(a) \mathrel{\widehat{=}} \begin{cases} a & \text{if } a \in C \\ l.a & \text{otherwise} \end{cases} \qquad r(a) \mathrel{\widehat{=}} \begin{cases} a & \text{if } a \in C \\ r.a & \text{otherwise} \end{cases} \qquad \begin{array}{rcl} c(a) & \mathrel{\widehat{=}} & a \ \text{ if } a \in C \\ c(l.a) & \mathrel{\widehat{=}} & a \ \text{ if } a \notin C \\ c(r.a) & \mathrel{\widehat{=}} & a \ \text{ if } a \notin C \end{array}$$

and

$$A \mathrel{\widehat{=}} l(\Sigma - C) \cup C \qquad B \mathrel{\widehat{=}} r(\Sigma - C) \cup C$$

and we assume $l(\Sigma) \cap C = r(\Sigma) \cap C = \{\}$.

In order to give a semantic definition to this operator, we first consider what offers P and Q perform when $P \underset{C}{\Vert} Q$ performs some offer v. By analogy with the definition of interleaving, we claim that P and Q perform $\underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q} v$ and $\underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q} v$, respectively, where the operators $\underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q}$ and $\underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q}$ are defined by

Definition 3.4.16: For all offers v and w, if

$$\exists\, v_P \in \text{items} \sqsubseteq_P ;\ v_Q \in \text{items} \sqsubseteq_Q \quad v_P \quad C = v_Q \quad C = v \quad C \wedge v_P \uplus (v_Q \setminus C) = v$$

then

$$\begin{aligned} \underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q} v &\mathrel{\widehat{=}} \sqcup_{\sqsubseteq_P} \{v_P \subseteq v \mid v_P \quad C = v \quad C \wedge v - (v_P \setminus C) \in \text{items} \sqsubseteq_Q\} \\ \underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q} v &\mathrel{\widehat{=}} v - (\underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q} v) \setminus C \end{aligned}$$

$\diamond$

P performs a subbag of v that contains all of $v \quad C$, and such that Q can perform the rest of v along with $v \quad C$; subject to these conditions, it performs the subbag of v that is maximal with respect to its offer relation. Q performs all of v except for those events outside the synchronisation set that are performed by P.

If P and Q have offer relations $\sqsubseteq_P$ and $\sqsubseteq_Q$ then $P \underset{C}{\Vert} Q$ will have offer relation $\sqsubseteq_P \underset{C}{\Vert} \sqsubseteq_Q$, defined by

Definition 3.4.17 (Sharing composition of offer relations) For all offers v and w, if

$$\begin{aligned} \exists\, v'_P, w'_P \in \text{items} \sqsubseteq_P ;\ v'_Q, w'_Q \in \text{items} \sqsubseteq_Q \quad & v'_P \quad C = v'_Q \quad C = v \quad C \\ & \wedge w'_P \quad C = w'_Q \quad C = w \quad C \\ & \wedge v'_P \uplus (v'_Q \setminus C) = v \\ & \wedge w'_P \uplus (w'_Q \setminus C) = w \end{aligned}$$

then

$$v(\sqsubseteq_P \underset{C}{\overleftarrow{\|}} \sqsubseteq_Q)w \Leftrightarrow v_P \sqsubset_P w_P \vee v_P = w_P \wedge v_Q \sqsubseteq_Q w_Q$$

where

$$v_P = \underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q} v \qquad w_P = \underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q} w \qquad v_Q = \underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q} v \qquad w_Q = \underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q} w$$

$\diamond$

$\sqsubseteq_P \underset{C}{\overleftarrow{\|}} \sqsubseteq_Q$ is the lexicographical ordering on the corresponding projections of its arguments. We can now give the semantics for the $\underset{C}{\overleftarrow{\|}}$ operator.

$$\mathcal{A}_{BT}\ P \underset{C}{\overleftarrow{\|}} Q\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq_P \underset{C}{\overleftarrow{\|}} \sqsubseteq_Q, s) \mid (\tau, \sqsubseteq_P, \underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q} s) \in A_P \wedge (\tau, \sqsubseteq_Q, \underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q} s) \in A_Q\}$$

where the $\underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q}$ and $\underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q}$ operators are extended to traces by

$$\underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q} s \mathrel{\widehat{=}} \{t \mapsto \underset{C}{\Uparrow}_{\sqsubseteq_P,\sqsubseteq_Q}(s \uparrow t) \mid t \in Is\} \qquad \underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q} s \mathrel{\widehat{=}} \{t \mapsto \underset{C}{\Downarrow}_{\sqsubseteq_P,\sqsubseteq_Q}(s \uparrow t) \mid t \in Is\}$$

We can define a right-biased communicating parallel operator by

$$P \underset{C}{\overrightarrow{\|}} Q \mathrel{\widehat{=}} Q \underset{C}{\overleftarrow{\|}} P$$

Note that if $\Sigma P \subseteq A$ and $\Sigma Q \subseteq B$ for some sets A and B such that $A \cap B = C$ then $P \underset{C}{\overleftarrow{\|}} Q = P\ {}^{A}\overleftarrow{\|}^{B}\ Q$ and $P \underset{C}{\overrightarrow{\|}} Q = P\ {}^{A}\overrightarrow{\|}^{B}\ Q$.

3.4.10 Nondeterministic choice

The process $P \sqcap Q$ either acts like P or like Q. Therefore the set of behaviours of $P \sqcap Q$ is the union of the behaviours of P and Q:

$$\mathcal{A}_{BT}\ P \sqcap Q\ \rho \mathrel{\widehat{=}} \mathcal{A}_{BT}\ P\ \rho \cup \mathcal{A}_{BT}\ Q\ \rho$$

The following laws hold for the nondeterministic choice operator:

Law 3.4.18 (Commutativity of nondeterministic choice) $P \sqcap Q = Q \sqcap P$. $\triangle$

Law 3.4.19 (Idempotence of nondeterministic choice) $P \sqcap P = P$. $\triangle$

Law 3.4.20 (Associativity of nondeterministic choice) $P \sqcap (Q \sqcap R) = (P \sqcap Q) \sqcap R$. $\triangle$

Law 3.4.21 (Distributivity) All operators except recursion distribute through nondeterministic choice:

Prefixing:	$a \xrightarrow{t} (P \sqcap Q) = a \xrightarrow{t} P \sqcap a \xrightarrow{t} Q$
External choice:	$P \,\square\, (Q \sqcap R) = P \,\square\, Q \sqcap P \,\square\, R$
	$(P \sqcap Q) \,\square\, R = P \,\square\, R \sqcap Q \,\square\, R$
Parallel composition:	$P \;{}^{X}\|^{Y}\; (Q \sqcap R) = P \;{}^{X}\|^{Y}\; Q \sqcap P \;{}^{X}\|^{Y}\; R$
	$(P \sqcap Q) \;{}^{X}\|^{Y}\; R = P \;{}^{X}\|^{Y}\; R \sqcap Q \;{}^{X}\|^{Y}\; R$
Interleaving:	$P \longleftarrow (Q \sqcap R) = P \longleftarrow Q \sqcap P \longleftarrow R$
	$(P \sqcap Q) \longleftarrow R = P \longleftarrow R \sqcap Q \longleftarrow R$
Hiding:	$(P \sqcap Q) \setminus X = P \setminus X \sqcap Q \setminus X$
Renaming:	$f(P \sqcap Q) = f(P) \sqcap f(Q)$
Sequential composition:	$(P \sqcap Q) \quad R = P \quad R \sqcap Q \quad R$
	$P \quad (Q \sqcap R) = P \quad Q \sqcap P \quad R$

and similar laws for the right biased operators. △

Infinite nondeterministic choice

The semantic definition for the infinite nondeterministic choice operator is similar:

$$\mathcal{A}_{BT} \quad {}_{i \in I} P_i \ \rho \mathrel{\widehat{=}} \bigcup \{\mathcal{A}_{BT} \ P_i \ \rho \mid i \in I\}$$

As in the Timed Failures Model, we need a restriction upon the sets of processes over which the choice can be made.

Definition 3.4.22: The set $\{P_i \mid i \in I\}$ is *uniformly bounded* if

$$\forall \tau \quad 0 \quad \exists\, n(\tau) \quad \forall\, i \in I \,;\, \rho \in ENV \quad (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT} \ P_i \ \rho \Rightarrow \# s \quad n(\tau)$$

and

$$\begin{aligned} &\forall \tau \quad 0 \quad \exists\, n(\tau) \quad \forall\, i \in I \,;\, \rho \in ENV \\ &\quad (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT} \ P_i \ \rho \Rightarrow \exists\, k \quad n(\tau) \,;\, J_0, \ldots, J_{k-1} : TINT \\ &\qquad J_0, \ldots, J_{k-1} \text{ partition } [0, \tau] \\ &\qquad \wedge \forall\, j : 0 \,..\, k-1 : t, t' : J_j \,;\, \chi, \psi : \mathrm{bag}\,\Sigma \quad (t, \chi) \sqsubseteq (t, \psi) \Leftrightarrow (t', \chi) \sqsubseteq (t', \psi) \end{aligned}$$

◇

These two conditions correspond to axioms B1 and B2 of the semantic space. The first condition states that there is a uniform bound on the number of events that any of the processes can perform within time τ; the second condition states that there is a uniform bound on the number of times that the offer relation can change shape within time τ.

The reader should be aware that this method does not always effectively model nondeterminism that does not manifest itself in a finite amount of time. For example, consider the process P_n that can perform n as:

$$P_0 \mathrel{\widehat{=}} STOP \qquad P_{n+1} \mathrel{\widehat{=}} a \xrightarrow{1} P_n$$

Let P be the process that chooses nondeterministically between the P_n:

$$P \mathrel{\widehat{=}} \quad {}_{n \in \mathbb{N}} P_n$$

P can perform any finite number of as. We would expect this to be different from the process P' that can perform an arbitrary number of as:

$$P' \mathrel{\widehat{=}} (a \xrightarrow{1} P') \sqcap STOP$$

However, our semantics gives the same value to both of these processes.

3.4.11 Hiding

In order to define the operation of hiding on processes we must first define hiding on offer relations. A bag of events w being offered by $P \setminus X$ corresponds to P offering a bag of events w' such that $w' \setminus X = w$. In general, P may be able to perform several bags w' such that $w' \setminus X = w$. We make the assumption that it performs the one that is maximal with respect to its offer relation. This can be thought of as a sort of maximal progress assumption in that the process performs as many hidden events as it wants.

We want an operator that, given an offer of $P \setminus X$, returns the corresponding offer of P. It will turn out that our semantic definition of renaming will be very similar to that for hiding, so we define an operator that can be used in both cases. The operator $\Uparrow^{g}_{\sqsubseteq} : OFF \nrightarrow OFF$ is such that $\Uparrow^{g}_{\sqsubseteq} w$ is the $\sqsubseteq$-strongest offer w' such that $gw' = w$:

$$\exists\, w' \in \text{items} \sqsubseteq \;\; gw' = w \qquad \Rightarrow \qquad \Uparrow^{g}_{\sqsubseteq} w = \sqcup_{\sqsubseteq}\{w' \in \text{items} \sqsubseteq \mid gw' = w\}$$

Hence, w being offered by $P \setminus X$ corresponds to $\Uparrow^{-\setminus X}_{\sqsubseteq} w$ being offered by P. The operator $\Uparrow^{-\setminus X}_{\sqsubseteq}$ can be though of as a sort of "inverse hiding" operator in the sense that $\_ \setminus X \circ \Uparrow^{-\setminus X}_{\sqsubseteq} = id$. The offer $\Uparrow^{-\setminus X}_{\sqsubseteq} w$ is the $\sqsubseteq$-maximal member of $(\_ \setminus X)^{-1}(w)$.

$P \setminus X$ will prefer w to v if P prefers $\Uparrow^{-\setminus X}_{\sqsubseteq} w$ to $\Uparrow^{-\setminus X}_{\sqsubseteq} v$. Hence we have the following definition for hiding on offer relations:

Definition 3.4.23 (Hiding on offer relations) For all offers v and w, if

$$\exists\, v', w' \in \text{items} \sqsubseteq \;\; v' \setminus X = v \wedge w' \setminus X = w$$

then

$$v(\sqsubseteq \setminus X)w \;\Leftrightarrow\; \Uparrow^{-\setminus X}_{\sqsubseteq} v \sqsubseteq \Uparrow^{-\setminus X}_{\sqsubseteq} w$$

◇

Note that $\mathrm{items}(\sqsubseteq \setminus X) = \{v \setminus X \mid v \in \mathrm{items} \sqsubseteq\}$.

An offer relation $\sqsubseteq$ of $P \setminus X$ must have resulted from an offer relation $\sqsubseteq'$ of P, such that $\sqsubseteq' \setminus X = \sqsubseteq$. Then for $P \setminus X$ to perform trace s, P must perform trace $\Uparrow_{\sqsubseteq'}^{-\setminus X} s$ where the $\Uparrow$ operator is defined on traces by

$$\Uparrow_{\sqsubseteq'}^{g} s = \{t \mapsto \Uparrow_{\sqsubseteq'}^{g}(s \uparrow t) \mid t \in Is\}$$

This exists only if for all t there is some $v \in \mathrm{items} \sqsubseteq'$ such that $v \setminus X = s \uparrow t$: this is equivalent to saying $\forall t \quad s \uparrow t \in \mathrm{items} \sqsubseteq$. Thus we have the following definition:

$$\mathcal{A}_{BT} \llbracket P \setminus X \rrbracket \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \mid \forall t \quad s \uparrow t \in \mathrm{items} \sqsubseteq \wedge \exists \sqsubseteq' \quad \sqsubseteq' \setminus X = \sqsubseteq \wedge (\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus X} s) \in A_P\}$$

The following laws relate to the hiding operator:

Law 3.4.24 (General laws for hiding) $P \setminus \{\} = P$ and $(P \setminus X) \setminus Y = P \setminus (X \cup Y)$. $\triangle$

Law 3.4.25 (Distribution of hiding) The following two laws hold:

$$(a \xrightarrow{t} P) \setminus X = \begin{cases} WAIT\ t\,; (P \setminus X) & \text{if } a \in X \\ a \xrightarrow{t} (P \setminus X) & \text{if } a \notin X \end{cases}$$

$$(P\ {}_A\!\|\!|_B\ Q) \setminus X = (P \setminus X)\ {}_A\!\|\!|_B\ (Q \setminus X) \text{ if } X \subseteq A \setminus B \cup B \setminus A$$

$\triangle$

3.4.12 Renaming

The definition of renaming is, in many ways, very similar to the definition of hiding. To define renaming on processes, we must first define renaming on offer relations. The process $g(P)$ performing v corresponds to P performing $\Uparrow_{\sqsubseteq}^{g} v$ (assuming of course that there is some $v' \in \mathrm{items} \sqsubseteq$ such that $gv' = v$). Hence the offer relation renaming operator, which we write as $\odot$, has the following definition:

Definition 3.4.26 (Renaming of offer relations) For all offers v and w, if

$$\exists v', w' \in \mathrm{items} \sqsubseteq \quad gv' = v \wedge gw' = w$$

then

$$v(g \odot \sqsubseteq) w \Leftrightarrow \Uparrow_{\sqsubseteq}^{g} v \sqsubseteq \Uparrow_{\sqsubseteq}^{g} w$$

$\diamond$

Note that $\text{items}(g \odot \sqsubseteq) = \{gv \mid v \in \text{items} \sqsubseteq\}$. We shall sometimes choose to write $g \odot \sqsubseteq$ as $g\sqsubseteq$.

A behaviour $(\tau, \sqsubseteq, s)$ of $g(P)$ must correspond to a behaviour $(\tau, \sqsubseteq', \Uparrow^{g}_{\sqsubseteq'} s)$ of P, such that $g \odot \sqsubseteq' = \sqsubseteq$. This is well defined only if $\forall t \quad s \uparrow t \in \text{items} \sqsubseteq$. Hence we have the following definition:

$$\mathcal{A}_{BT}\ g(P)\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \mid \forall t \quad s \uparrow t \in \text{items} \sqsubseteq \wedge \exists \sqsubseteq' \quad g \odot \sqsubseteq' = \sqsubseteq \wedge (\tau, \sqsubseteq', \Uparrow^{g}_{\sqsubseteq'} s) \in A_P\}$$

For bijective g , we have

$$\mathcal{A}_{BT}\ g(P)\ \rho \mathrel{\widehat{=}} \{(\tau, g \odot \sqsubseteq, gs) \mid (\tau, \sqsubseteq, s) \in A_P\}$$

Where in this case $g \odot \sqsubseteq = \{(gv, gw) \mid v \sqsubseteq w\}$.

The following laws hold for the renaming operator:

Law 3.4.27 (Successive renaming) $f(g(P)) = (f \quad g)(P)$. $\triangle$

Law 3.4.28 (Distribution of renaming) $g(a \longrightarrow P) = ga \longrightarrow g(P)$. $\triangle$

Law 3.4.29 (Distribution of renaming by bijective functions) If g is a bijection then the following distribution laws hold :

$$\begin{aligned}
g(P\ {}^{X}\|\!\|^{Y}\ Q) &= g(P)\ {}^{gX}\|\!\|^{gY}\ g(Q) \\
g(\mu X \quad FX) &= \mu Y \quad g(F(g^{-1} Y)) \\
g(P \setminus X) &= g(P) \setminus gX \\
g(P \sqsupset Q) &= g(P) \sqsupset g(Q) \\
g(P \longleftarrow Q) &= g(P) \longleftarrow g(Q)
\end{aligned}$$

$\triangle$

3.4.13 Sequential composition

A behaviour $(\tau, \sqsubseteq, s)$ of $P \quad Q$ can come about in three ways:

- a behaviour of P that does not terminate before time τ;
- a behaviour of P that terminates between times $\tau - \delta$ and τ; *or*
- a behaviour of P that terminates successfully before time $\tau - \delta$ followed by a behaviour of Q.

Note that we have to hide the event from any behaviour of P in order to make sure that it happens (silently) as soon as possible. We have the following definition:

$$
\begin{aligned}
&\mathcal{A}_{BT}\ P\ \ Q\ \rho \mathrel{\widehat{=}} \\
&\quad \{(\tau, \sqsubseteq_P, s_P) \mid \forall t \quad s_P \uparrow t \in \text{items} \sqsubseteq_P \\
&\qquad\qquad\qquad \wedge \exists \sqsubseteq'_P \quad \sqsubseteq'_P \setminus \quad = \sqsubseteq_P \wedge (\tau, \sqsubseteq'_P, \Uparrow^{\setminus\checkmark}_{\sqsubseteq'_P} s_P) \in A_P \wedge \quad \notin \Sigma(\Uparrow^{\setminus\checkmark}_{\sqsubseteq'_P} s_P)\} \\
&\quad \cup\{(\tau, \sqsubseteq_P \quad (t, \tau] \otimes \langle\{\!|\!\}\rangle, s_P) \mid \\
&\qquad t \quad \tau < t + \delta \wedge \forall t' \quad s_P \uparrow t' \in \text{items}(\sqsubseteq_P \quad (t, \tau] \otimes \langle\{\!|\!\}\rangle) \\
&\qquad \wedge \exists \sqsubseteq'_P \quad \sqsubseteq'_P \setminus \quad = \sqsubseteq_P \wedge (t, \sqsubseteq'_P, \Uparrow^{\setminus\checkmark}_{\sqsubseteq'_P} s_P) \in A_P \wedge begin((\Uparrow^{\setminus\checkmark}_{\sqsubseteq'_P} s_P) \quad) = t\} \\
&\quad \cup\{(\tau, \sqsubseteq_P \quad (t, t + \delta) \otimes \langle\{\!|\!\}\rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad s_Q + t + \delta) \mid \\
&\qquad t \quad \tau - \delta \wedge \forall t' \quad s_P \uparrow t' \in \text{items} \sqsubseteq_P \\
&\qquad \wedge \exists \sqsubseteq'_P \quad \sqsubseteq'_P \setminus \quad = \sqsubseteq_P \wedge (t, \sqsubseteq'_P, \Uparrow^{\setminus\checkmark}_{\sqsubseteq'_P} s_P) \in A_P \wedge begin((\Uparrow^{\setminus\checkmark}_{\sqsubseteq'_P} s_P) \quad) = t \\
&\qquad \wedge (\tau - (t + \delta), \sqsubseteq_Q, s_Q) \in A_Q\}
\end{aligned}
$$

We have the following laws for sequential composition:

Law 3.4.30 (Associativity of sequeutial composition) $(P \quad Q) \quad R = P \quad (Q \quad R)$ and $(a \longrightarrow P) \quad Q = a \longrightarrow (P \quad Q)$. $\triangle$

Law 3.4.31 (*STOP* is a left zero of sequential composition) $STOP \quad P = STOP$. $\triangle$

3.4.14 Delay

Consider a behaviour $(\tau, \sqsubseteq, s)$ of $WAIT\ t\ ; P$:

- if $t > \tau$ then the process can perform and offer nothing;
- if $t \quad \tau$ then the process can act like P, temporally shifted by t.

This gives the following definition:

$$
\begin{aligned}
\mathcal{A}_{BT}\ \ WAIT\ t ; P\ \ \rho \mathrel{\widehat{=}} &\{(\tau, [0, \tau] \otimes \langle\{\!|\!\}\rangle, \{\!|\!\}) \mid t > \tau\} \\
&\cup\{(\tau, [0, t) \otimes \langle\{\!|\!\}\rangle \quad \sqsubseteq + t, \prec\succ \quad s + t) \mid t \quad \tau \wedge (\tau - t, \sqsubseteq, s) \in A_P\}
\end{aligned}
$$

The following laws hold:

Law 3.4.32 (Effect of *SKIP*) $SKIP \quad P = WAIT\ \delta\ ; P$. $\triangle$

Law 3.4.33 (Successive delays) $WAIT\ t ; WAIT\ t' ; P = WAIT\ t + t'\ ; P$. $\triangle$

3.4.15 Timeout

In [Sch90], Schneider defines a timeout operator by

$$P \;{}^{t}\; Q \mathrel{\widehat{=}} (P \quad WAIT\ t\,;\, trig \longrightarrow Q) \setminus trig$$

where *trig* is an event not in the alphabets of P or Q. This begins by acting like P; if no visible event has occurred by time t then the process times out by performing the event *trig* silently, and after a delay of length δ acts like Q. If P is able to perform its first visible event precisely at time t, then it is nondeterministic whether or not the timeout occurs.

We will define our timeout operator by refining the external choice in the process definition to either a left- or right-biased choice. We consider the effects of these two different choices.

- If we choose a left-biased choice, then if the process P is willing to do its first event precisely at time t, then that event is offered stronger than the silent *trig*, and so will occur if the environment is willing to perform it.
- If we choose a right-biased choice, then if the process P is willing to do its first event precisely at time t, then that event is offered weaker than the silent *trig*, and so will occur only if the environment is *not* willing to idle.

The first choice seems to be more useful in practice. The timeout operator is often used where the process is initially waiting for an event to be offered by the environment; if the event is not offered within a certain time then it times out and acts accordingly. It seems sensible to give the environment as much chance as possible to respond; we therefore specify that $P \;{}^{t}\; Q$ will be willing to accept the events of P at all times up to *and including* t.

The first choice also produces the simpler operator: it turns out that with this choice the offer relation of $P \;{}^{t}\; Q$ at time t is simply the offer relation of P at that time; if we were to make the second choice then the offer relation would be somewhat more complicated.

We therefore have the following definition:

Definition 3.4.34 (Timeout) The process $P \;{}^{t}\; Q$ is defined by

$$P \;{}^{t}\; Q \mathrel{\widehat{=}} (P \mathrel{\boxed{\square}} WAIT\ t\,;\, trig \longrightarrow Q) \setminus trig$$

where *trig* is an event not in the alphabets of P and Q. ◇

We can use this definition to give a semantic equation for the timeout operator.

$$\begin{aligned} \mathcal{A}_{BT}\ P \;{}^{t}\; Q\ \rho = {} & \{(\tau, \sqsubseteq, s) \mid (\tau \quad t \vee begin\ s \quad t) \wedge (\tau, \sqsubseteq, s) \in A_P\} \\ & \cup \{(\tau, \sqsubseteq \quad (t, \tau] \otimes \langle\!\{\!\}\!\rangle, \prec\succ) \mid t < \tau < t + \delta \wedge (t, \sqsubseteq, \prec\succ) \in A_P\} \\ & \cup \{(\tau, \sqsubseteq_P \quad (t, t+\delta) \oslash \langle\!\{\!\}\!\rangle \quad \sqsubseteq_Q + t + \delta, \prec\succ \quad s_Q + t + \delta) \mid \\ & \qquad \tau \quad t + \delta \wedge (t, \sqsubseteq_P, \prec\succ) \in A_P \wedge (\tau - t - \delta, \sqsubseteq_Q, s_Q) \in A_Q\} \end{aligned}$$

A behaviour of $P \;{}^{t}\; Q$ can either be:

- a behaviour of P that either ends before time t or where a visible event has occurred by time t;

- a behaviour of P in which no visible events are observed up until time t, followed by a short period during which control is being transferred to Q; *or*
- a behaviour of P in which no visible events are observed up until time t, followed by a short delay, followed by a behaviour of Q.

3.4.16 Timed transfer

The process $P \underset{t}{\ } Q$ acts like P up until time t, at which time control is passed to Q (after a short delay) regardless of the progress made by P. This differs slightly from the definition given in [Sch90], where control was not transferred to Q if P terminated normally before time t. The semantic definition is as follows.

$$\begin{aligned}
\mathcal{A}_{BT}\ P \underset{t}{\ } Q\ \rho \mathrel{\widehat{=}} & \{(\tau, \sqsubseteq, s) \mid \tau \quad t \wedge (\tau, \sqsubseteq, s) \in A_P\} \\
& \cup \{(\tau, \sqsubseteq \quad (t, \tau] \otimes \langle \{\!|\,|\!\} \rangle, s \quad \prec\succ) \mid t < \tau < t + \delta \wedge (t, \sqsubseteq, s) \in A_P\} \\
& \cup \{(\tau, \sqsubseteq_P \quad (t, t + \delta) \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad \prec\succ \quad s_Q + t + \delta) \mid \\
& \qquad \tau \quad t + \delta \wedge (t, \sqsubseteq_P, s_P) \in A_P \wedge (\tau - t - \delta, \sqsubseteq_Q, s_Q) \in A_Q\}
\end{aligned}$$

A behaviour of $P \underset{t}{\ } Q$ can be either:

- a behaviour of P that ends no later than t;
- a behaviour of P up to time t, followed by a short delay during which control is being transfered to Q; *or*
- a behaviour of P up to time t, followed by a short delay, followed by a behaviour of Q starting at time $t + \delta$.

3.4.17 Interrupts

The process $P \underset{i}{\nabla} Q$ initially acts like P except it is always willing to perform the interrupt event i. If an i occurs, control is passed to the interrupt handler Q, after a delay of length δ. We assume that P cannot perform the event i — it cannot interrupt itself.

Before the event i occurs, the process should always offer i; it should be willing to perform an i *in addition* to whatever actions P offers. For example, if P has an offer relation $\sqsubseteq$ with $(t, \{\!| b |\!\}) \sqsupset (t, \{\!| a |\!\}) \sqsupset (t, \{\!|\,|\!\})$, then $P \underset{i}{\nabla} Q$ should have an offer relation $\sqsubseteq'$ with

$$(t, \{\!| i, b |\!\}) \sqsupset' (t, \{\!| i, a |\!\}) \sqsupset' (t, \{\!| i |\!\}) \sqsupset' (t, \{\!| b |\!\}) \sqsupset' (t, \{\!| a |\!\}) \sqsupset' (t, \{\!|\,|\!\})$$

In general, if P has offer relation $\sqsubseteq$, then before the interrupt occurs $P \underset{i}{\nabla} Q$ should have offer relation $\sqsubseteq \oplus i$ given by

$$\sqsubseteq \oplus i \mathrel{\widehat{=}} \sqsubseteq \longrightarrow (I \sqsubseteq \otimes \langle \{\!| i |\!\}, \{\!|\,|\!\} \rangle)$$

The semantic definition of $P \underset{i}{\nabla} Q$ is

$$
\begin{aligned}
&\mathcal{A}_{BT}\ P \underset{i}{\nabla} Q\ \rho \mathrel{\widehat{=}} \\
&\quad \{(\tau, \sqsubseteq \oplus\, i, s) \mid (\tau, \sqsubseteq, s) \in A_P \wedge i \notin \Sigma s\} \\
&\quad \cup \{(\tau, \sqsubseteq \oplus\, i \quad (t, \tau] \otimes \langle\{\!|\,|\!\}\rangle, s \quad \prec(t, i)\succ) \mid t \quad \tau < t + \delta \wedge i \notin \Sigma s \wedge (t, \sqsubseteq, s) \in A_P\} \\
&\quad \cup \{(\tau, \sqsubseteq_P \oplus\, i \quad (t, t+\delta) \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad \prec(t, i)\succ \quad s_Q + t + \delta) \mid \\
&\qquad\quad \tau \quad t + \delta \wedge i \notin \Sigma s_P \wedge (t, \sqsubseteq_P, s_P) \in A_P \wedge (\tau - t - \delta, \sqsubseteq_Q, s_Q) \in A_Q\}
\end{aligned}
$$

A behaviour of $P \underset{i}{\nabla} Q$ can be either:

- a behaviour of P where an additional i is offered at all times, and no i occurs;
- a behaviour of P where an additional i is offered at all times and an i first occurs at time t, followed by a short pause during which control is being transferred to Q; *or*
- a behaviour of P where an additional i is offered at all times and an i first occurs at time t, followed by a behaviour of Q after a delay of length δ.

3.4.18 Recursion

In order to define recursion, we first define a metric on the space $\mathcal{M}_{TB}$. We do this by considering the first time at which two processes may be distinguished. We define an operator on behaviour sets which gives the behaviour of a process up to a certain time.

$$A \quad t \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \in A \mid \tau \quad t\}$$

We define the metric on $\mathcal{M}_{TB}$ by

$$d(A_P, A_Q) \mathrel{\widehat{=}} \inf(\{2^{-t} \mid A_P \quad t = A_Q \quad t\} \cup \{1\})$$

The semantics of a BTCSP term P is a function of the free variables appearing in the definition of P. If P is the body of a recursive process, then the recursion is well defined if P corresponds to a contraction mapping in $\mathcal{M}_{TB}$. For this to be true it is sufficient for P to be *constructive* for the bound variable.

Constructive processes

We define constructive terms as follows:

Definition 3.4.35: Term P is *t-constructive* for X iff

$$\forall t_0 : T\,; \rho : ENV \quad \mathcal{A}_{BT}\ P\ \rho \quad t_0 + t = \mathcal{A}_{BT}\ P\ \rho[\rho\ X \quad t_0/X] \quad t_0 + t$$

$\diamond$

P is t-constructive for X if its behaviour up until time $t_0 + t$ is independent of the value of X after t_0.

Definition 3.4.36: Term P is *constructive* for X iff there is some strictly positive t such that P is t-constructive for X. $\diamond$

From the semantic equations for the BTCSP operators we can derive a number of results about constructive terms.

Lemma 3.4.37: For any X and t,

1. $STOP$, $SKIP$ and $WAIT\ t'$ are t-constructive for X;
2. X is 0-constructive for X, and t-constructive for $Y \neq X$;
3. $\mu X \quad P$ is t-constructive for X

$\heartsuit$

Lemma 3.4.38: If P is t-constructive for X then

1. $a \xrightarrow{t'} P$ and $WAIT\ t'\,;P$ are $t + t'$-constructive for X;
2. $\mu Y \quad P$, $P \setminus A$ and $f(P)$ are t-constructive for X;
3. P is t'-constructive for X, for any $t' < t$.

$\heartsuit$

Lemma 3.4.39: If P is t_1-constructive for X and Q is t_2-constructive for X then

1. $P \boxdot Q$, $P \Box Q$, $P\ {}_p\sqcap_q\ Q$, $P \overleftarrow{\|}\ Q$, $P \overrightarrow{\|}\ Q$, $P\ {}^A\overleftarrow{\|}^B\ Q$, $P\ {}^A\overrightarrow{\|}^B\ Q$, $P \longleftarrow Q$ and $P \longrightarrow Q$ are all $t_1 \sqcap t_2$-constructive for X;
2. $P \quad Q$ is $t_1 \sqcap t_2 + \delta$-constructive for X.

$\heartsuit$

Recursive processes

The semantics of a term P with free variable X may be thought of as a function of the semantics of X; it is the function that associates with each member Y of the semantic space S_{TB}, the value of P evaluated in an environment where X is bound to Y. We represent this function by $M(X, P)\rho$:

$$M(X,P)\rho \mathrel{\widehat{=}} \lambda\, Y \quad \mathcal{A}_{BT}\ P\ \rho[Y/X]$$

Note that the environment ρ supplies the bindings for any variables other than X. We use this mapping to give a semantics to the immediate recursion operator:

$$\mathcal{A}_{BT}\ \mu\, X \quad P\ \rho \mathrel{\widehat{=}} \text{the nnique fixed point of the mapping } M(X,P)\rho$$

We will show that if P is constructive for X then the mapping $M(X, P)\rho$ is a contraction mapping, and hence the semantics is well defined.

Lemma 3.4.40: If term P is constructive for variable X, then $M(X, P)\rho$ is a contraction mapping on the semantic space S_{TB}. $\heartsuit$

Proof: Let $F \mathrel{\hat{=}} M(X, P)\rho$. F is a contraction mapping iff

$$\exists\, r < 1 \quad \forall S, T : \mathcal{S}_{TB} \quad d(F(S), F(T)) \quad r.d(S, T)$$

where d is the metric defined by

$$d(S, T) \mathrel{\hat{=}} \inf(\{2^{-t} \mid S \quad t = T \quad t\} \cup \{1\})$$

Pick S and T in $\mathcal{S}_{TB}$. If $S = T$ then both sides of the above equation are zero. Otherwise, let $d(S, T) = 2^{-t}$. Now F is constructive, so there is a strictly positive t' such that

$$S \quad t = T \quad t \Rightarrow F(S) \quad t + t' = F(T) \quad t + t'$$

so

$$d(F(S), F(T)) \quad 2^{-(t+t')} = 2^{-t'}.d(S, T)$$

Hence F is a contraction mapping because $2^{-t'} < 1$. □

We have shown that the mapping corresponding to a constructive term is a contraction mapping on $\mathcal{S}_{TB}$. To show that it has a unique fixed point, we require the following result from [Sut75]:

Theorem 3.4.41 (Banach's Fixed Point Theorem) Let (M, d) be a complete metric space, and let $F : M \to M$ be a contraction mapping. Then F has a unique fixed point $\mathit{fix}(F)$. Furthermore, for all $S \in M$ we have $\mathit{fix}(F) = \lim_{n\to\infty} F^n(S)$. ♡

In order to apply this, we need the following lemma.

Lemma 3.4.42: $\mathcal{M}_{TB}$ is a complete subspace of $\mathcal{S}_{TB}$. ♡

Proof of lemma: For all $n \in \;$, let A_n be a member of $\mathcal{M}_{TB}$, and let $\langle A_n \mid n \in \; \rangle$ have limit A. We must show $A \in \mathcal{M}_{TB}$. Let $d_n \mathrel{\hat{=}} d(A_n, A)$. Then $d_n \to 0$, so $\forall \tau \quad \exists N_\tau \quad \forall n \quad N_\tau \quad d_n < 2^{-\tau}$, i.e.

$$\forall n \quad N_\tau \quad A_n \quad \tau = A \quad \tau$$

The axioms of $\mathcal{M}_{TB}$ can now easily be proved. We prove axiom B5 for illustration. Let $(\tau, \sqsubseteq, s) \in A$, $\tau' > \tau$, $\Omega \in \mathit{EOFF}$ such that $I\Omega = (\tau, \tau']$. Let $n \quad N_{\tau'}$, so $A \quad \tau' = A_n \quad \tau'$. Then $(\tau, \sqsubseteq, s) \in A_n$ and since $A_n \in \mathcal{M}_{TB}$ there is some $\sqsubseteq'$ such that $\sqsubseteq' \quad \tau = \sqsubseteq$ and $(\tau', \sqsubseteq', s \quad \sqcup_{\sqsubseteq' \upharpoonright \tau} \Omega) \in A_n$. Hence $(\tau', \sqsubseteq', s \quad \sqcup_{\sqsubseteq' \upharpoonright \tau} \Omega) \in A$ as required. □

Corollary 3.4.43: If $F : \mathcal{S}_{TB} \to \mathcal{S}_{TB}$ is a contraction mapping that maps $\mathcal{M}_{TB}$ into itself, then F has a unique fixed point, which lies in $\mathcal{M}_{TB}$. ♡

Proof: This follows immediately from Banach's fixed point theorem, because a contraction mapping on $\mathcal{S}_{TB}$ is a contraction mapping on $\mathcal{M}_{TB}$. □

Lemma 3.4.40 and corollary 3.4.43 can be combined to give the following result:

Theorem 3.4.44: If term P is constructive for variable X, then the semantics for $\mathcal{A}_{BT}\ \mu X \quad P\ \rho$ is well defined in all environments ρ. ♡

The semantic definition gives rise to the following equivalence.

Theorem 3.4.45: $\mu X \quad P = P[\mu X \quad P/X]$ ♡

This result justifies the use of recursive equations, such as $X \mathrel{\widehat{=}} a \xrightarrow{I} X$, as process definitions.

Delayed recursion

To give a semantics to the delayed recursion operator, we consider the composition of the mapping $M(X,P)\rho$ with the function W_δ which delays its argument by δ.

Definition 3.4.46: If P is a term, X is a variable, and Y is a member of $\mathcal{S}_{TB}$, then

$$M_\delta(X,P)\rho \mathrel{\widehat{=}} M(X,P)\rho \circ W_\delta \quad \text{where} \quad W_\delta \mathrel{\widehat{=}} \lambda\, Y \quad \mathcal{A}_{BT}\ \mathit{WAIT}\ \delta\, ; X\ \rho[Y/X]$$

◇

The W_δ reflects the delay associated with this recursion. We may now give semantics for the delayed recursion operator:

$$\mathcal{A}_{BT}\ \mu X \quad P\ \rho \mathrel{\widehat{=}} \text{the unique fixed point of the mapping } M_\delta(X,P)\rho$$

It is very easy to prove that $M_\delta(X,P)\rho$ is always a contraction mapping, and hence delayed recursion is always well defined.

Mutual recursion

We can define a BTCSP term in terms of a vector of mutually recursive equations: $\langle X_i = P_i \mid i \in I\rangle_j$ represents the jth component of the vector of terms defined by the set of equations $\{X_i = P_i\}$. We shall write $\underline{P}$ for the vector $\langle P_i \mid i \in I\rangle$, etc.

In order to give semantics to $\langle X_i = P_i \mid i \in I\rangle_j$ we consider the semantic domain $\mathcal{S}_{TB}^I$, i.e. the product space with one copy of $\mathcal{S}_{TB}$ for each element of I. We define a metric on this space by

$$\underline{d}(\underline{U},\underline{V}) \mathrel{\widehat{=}} \sup\{d(U_i, V_i) \mid i \in I\}$$

If $\underline{P}$ is a vector of BTCSP terms, $\underline{X}$ is a vector of variables, and $\underline{Y}$ a vector of members of $\mathcal{S}_{TB}$, all indexed by set I, then the mapping on $\mathcal{S}_{TB}^I$ corresponding to $\underline{P}$ is given by

$$M(\underline{X},\underline{P})\rho \mathrel{\widehat{=}} \lambda\, \underline{Y} \quad \mathcal{A}_{BT}\ \underline{P}\ \rho[Y_i/X_i \mid i \in I]$$

We can now give a semantics to mutual recursion.

Definition 3.4.47: If $\underline{P}$ is a vector of BTCSP terms, then

$$\mathcal{A}_{BT}\ \langle X_i = P_i \mid i \in I\rangle_j\ \rho \mathrel{\widehat{=}} S_j \text{ where } \underline{S} \text{ is a fixed point of } M(\underline{X},\underline{P})\rho$$

◇

This is well-defined when all fixed points of $M(\underline{X}, \underline{P})\rho$ agree on the jth component. In the rest of this section, we study under what circumstances $M(\underline{X}, \underline{P})\rho$ has a *unique* fixed point. We will need some definitions relating to partial orders.

Definition 3.4.48: A partial order $\prec$ on a set S is a well-ordering if there is no infinite descending chain $\langle s_i \mid i \in \ \rangle$ such that $s_{i+1} \prec s_i$, for all $i \in$. The initial segment of s in $(S, \prec)$ is the set of elements less than s; i.e. $\mathrm{seg}(s) \mathrel{\widehat{=}} \{s' \in S \mid s' \prec s\}$. ◇

We can now define what it means for a vector of terms $\underline{P}$ to be *constructive* for a vector of variables $\underline{X}$. This will turn out to be a sufficient condition for the existence of a unique fixed point.

Definition 3.4.49: A vector of terms $\langle P_i \mid i \in I\rangle$ is *t-constructive* for vector of variables $\langle X_i \mid i \in I\rangle$ if there is a well-ordering $\prec$ on I such that

$$\forall i, j : I \quad j \notin seg(i) \Rightarrow P_i \ t\text{-constructive for } X_j$$

◇

Definition 3.4.50: A vector of terms $\underline{P}$ is *constructive* for vector of variables $\underline{X}$ if there is a strictly positive t such that $\underline{P}$ is t-constructive for $\underline{X}$. ◇

If $\underline{P}$ is constructive for $\underline{X}$ then all unguarded recursive calls from term P_i are to a variable X_j such that $j \prec i$. Any sequence of unguarded recursive calls must correspond to a decreasing sequence of I, and so must be finite.

It will normally be possible to show that *all* terms P_i are constructive for *all* variables X_j.

Definition 3.4.51: A vector of terms $\langle P_i \mid i \in I\rangle$ is *uniformly t-constructive* for the vector of variables $\langle X_i \mid i \in I\rangle$ if P_i is t-constructive for X_j, for all i and j in I. ◇

Definition 3.4.52: A vector of terms $\underline{P}$ is *uniformly constructive* for vector of variables $\underline{X}$ if there is some strictly positive t such that $\underline{P}$ is uniformly t-constructive for $\underline{X}$. ◇

Lemma 3.4.53: If $\underline{P}$ is uniformly constructive for $\underline{X}$, then $\underline{P}$ is constructive for $\underline{X}$. ♡

We can now state the following theorem:

Theorem 3.4.54: If vector of terms $\underline{P}$ is constructive for vector of variables $\underline{X}$, then the mapping $M(\underline{X}, \underline{P})\rho$ has a unique fixed point in S_{TB}^{I}. ♡

Proof: The proof of this theorem follows closely the work of Davies and Schneider [DS90] and was given in [Low92a]; the interested reader should refer to that paper for details. The proof proceeded as follows: we defined a second vector of terms $\underline{Q}$ by

$$Q_i \mathrel{\widehat{=}} P_i[Q_j/X_j \mid j \in seg(i)]$$

we showed that this vector is well defined; we showed that the corresponding mapping $M(\underline{X}, \underline{Q})\rho$ is a contraction mapping and so has a unique fixed point; we showed that this fixed point is also a fixed point of $M(\underline{X}, \underline{P})\rho$; we showed that this fixed point is unique. □

From this theorem, we can deduce the following corollary:

Corollary 3.4.55: If vector of processes P is constructive for vector of variables X, then the semantics of $\langle X_i = P_i \rangle_i$ is well defined. ♡

3.5 Communication over channels

In the final two sections of this chapter we consider two variations on the Prioritized Model. In this section we consider how we can model the communication of values over channels; in the next section we consider what happens when we remove the nondeterministic choice operator from the syntax of the language.

Some models of concurrency have modelled communication by considering communications of different values to be fundamentally different events. For example if c is a channel inputting integers, then the events $c?1$ and $c?2$ would be treated as completely different. This is not adequate in a model where we want to place priorities upon actions: we do not want to have to make arbitrary decisions such as specifying that the process would prefer to input a 1 than a 2. We want to model the fact that processes have no preference as to which event they input along a channel. We will therefore arrange that the offer relation just records the fact that the process is willing to input *something* on a channel and says nothing about the values passed.

Another problem arises from processes such as

$$(c!1 \xrightarrow{1} P \longleftarrow c!2 \xrightarrow{1} P') \; {}^{X}\|\!\!\|^{Y} \; (c?x \xrightarrow{1} Q(x) \longleftarrow c?y \xrightarrow{1} Q'(y))$$

Here it is impossible to tell which process on the right inputs the 1 and which inputs the 2. To overcome this we shall insist that no process tries to write two things onto the same channel simultaneously or tries to read two things from the same channel simultaneously. This seems a reasonable assumption when one considers the physical nature of the channels: no wire can pass two messages simultaneously. It will be a requirement of anyone writing a process definition in BTCSP to check that this condition is satisfied. Fortunately the following lemma simplifies this.

Lemma 3.5.1: If a process P is such that

- no interleaving within the definition of the process has both sides able to write to the same channel, or has both sides able to read from the same channel; *and*
- all renaming within the process definition is one–one on channel names

then P does not try to write two things to any channel simultaneously or try to read two things from a channel simultaneously. ♡

One further problem arises from hiding of input channels. In normal Timed CSP we have the identity

$$(c?x : X \xrightarrow{1} P_x) \setminus c = WAIT\ 1 \ ; \quad {}_{x \in X} P_x$$

However, when we come to extend our model to include probabilities we will want to avoid such processes, because we will be unable to assign probabilities to the nondeterministic choice on the right. We therefore will not allow input channels to be hidden: again this seems a reasonable assumption.

We let *CHAN* be the set of all channel names. If *type*(c) is the type of data transmitted over channel c then we insist that

$$\forall\, c : CHAN\, ;\, x : type(c) \quad c.x \in \Sigma$$

i.e. all communications are visible events. We will write $c?x$ to represent the input of value x on channel c.

We define an *action* to be a pair consisting of a bag of events and a set of channels.

Definition 3.5.2: The set ACT of actions is defined by

$$ACT \mathrel{\widehat{=}} \mathrm{bag}\,\Sigma \times \quad (CHAN)$$

$\diamond$

We will write α, β for typical members of ACT, χ, ψ for typical members of bag Σ and ζ, η for typical members of $\quad (CHAN)$. The pair (χ, ζ) will represent the performance of the events of χ, and the input of events from the channels of ζ. We can now define the space *OFF* of offers, which are basically timed actions.

Definition 3.5.3: The set *OFF* of offers is defined by

$$OFF \mathrel{\widehat{=}} TIME \times ACT$$

$\diamond$

We will write v, w for typical members of *OFF*. The pair $(t, (\chi, \zeta))$ will represent that the process is willing to perform the events of χ and input on the channels of ζ at time t. So, for example, we will write $(3, (\{\!|a, a, b|\!\}, \{c, d\}))$ to represent the willingness of a process to perform two as and a b, and to input on channels c and d at time 3. For ease of notation, we will often write the elements of a particular action within bag brackets, marking input channels with a '?'; so, for example, we will write the above offer as $(3, \{\!|a, a, b, c?, d?|\!\})$.

As in the model without communication, we can now define the space *OFFREL* of offer relations as being those relations $\sqsubseteq$ of type $OFF \times OFF$ satisfying

1. $(t, \alpha) \sqsubseteq (t', \beta) \Rightarrow t = t'$ (comparable events occur at the same time);
2. $w \sqsubseteq w' \wedge w' \sqsubseteq w'' \Rightarrow w \sqsubseteq w''$ (transitivity);
3. $w \sqsubseteq w' \wedge w' \sqsubseteq w \Rightarrow w = w'$ (antisymmetry);
4. $w \in \mathrm{items} \sqsubseteq\ \Rightarrow w \sqsubseteq w$ (reflexivity on items$\sqsubseteq$);
5. $(t, \alpha), (t, \beta) \in \mathrm{items} \sqsubseteq\ \Rightarrow (t, \alpha) \sqsubseteq (t, \beta) \vee (t, \beta) \sqsubseteq (t, \alpha)$ (totality on items$\sqsubseteq$)

where $\mathrm{items} \sqsubseteq\ \mathrel{\widehat{=}} \{w \mid \exists\, v \quad v \sqsubseteq w \vee w \sqsubseteq v\}$.

Similarly, we now define the space TT of timed traces by

$$TT \mathrel{\widehat{=}} \{s : TIME \nrightarrow ACT \mid \exists\, \tau \quad \mathrm{dom}\, s = [0, \tau]\}$$

i.e. functions from times to actions.

Our semantic model for our language remains largely unchanged by this extension, the only change being the new definition of the space *OFFREL*.

We can now define a new operator, prefix choice. The process $c?d : D \xrightarrow{0} P_d$ is willing to input any value d on channel c, and then act like the process P_d, where P_d will, in general, depend on the value d input. In order to fit with our intuitions about causality, we will insist that the processes P_d are unable to perform any events at time 0. A behaviour of $c?d : D \xrightarrow{0} P_d$ will be either:

- a behaviour where nothing is performed, and the process is willing to input on channel c at any time; *or*
- a behaviour where an element $\hat{d}$ of D is input at time t, and the process then acts like $P_{\hat{d}}$.

This gives the following definition.

$$\begin{aligned} \mathcal{A}_{BT}\ c?d : D \xrightarrow{0} P_d\ \rho \mathrel{\widehat=} & \{(\tau, [0,\tau] \otimes \langle\{\!|c?|\!\}, \{\!|\!\}\rangle, \prec\succ) \mid \tau \in TIME\} \\ & \cup\{(\tau, [0,t] \otimes \langle\{\!|c?|\!\}, \{\!|\!\}\rangle \quad \sqsubseteq + t, (t, c?\hat{d}) \quad s + t) \mid \\ & \qquad \hat{d} \in D \wedge t \quad \tau \wedge (\tau - t, \{0\} \otimes \langle\{\!|\!\}\rangle \quad \sqsubseteq, s) \in \mathcal{A}_{BT}\ P_{\hat{d}}\ \rho\} \end{aligned}$$

This definition is well defined (i.e. it satisfies the healthiuess conditions of the semantic space) if the set of processes is uniformly bounded in the sense of section 3.4.10.

We can now define the general prefix choice operator. The process $c?d : D \xrightarrow{t_d} P_d$ inputs a value d on channel c, and then acts like process P_d after a delay of length t_d, where t_d may depend on the value d input. We can define this process by

$$c?d : D \xrightarrow{t_d} P_d \mathrel{\widehat=} c?d : D \xrightarrow{0} (WAIT\ t_d\ ; P_d)$$

3.6 A deterministic language and model

In this section we show how we can produce a completely deterministic language by removing the nondeterministic choice operator from the syntax of BTCSP. The results of this section also say something about the Prioritized Model: if a process in BTCSP is constructed without using the nondeterministic choice operator then it is deterministic in the following sense: if we know what the environment offers then there is only one way that the process can behave.

We define the syntax of Deterministic Timed CSP (DTCSP) by

$$
\begin{array}{rll}
P ::= & STOP \mid SKIP \mid WAIT\ t \mid X \mid & \text{basic processes} \\
& a \xrightarrow{t} P \mid P \quad P \mid WAIT\ t\,; P \mid & \text{sequential composition} \\
& P \mathbin{\langle\!\Box} P \mid P \mathbin{\Box\!\rangle} P \mid c?a : A \xrightarrow{t_a} P_a \mid & \text{alternation} \\
& P \mathbin{\langle\!\|} P \mid P \mathbin{\|\!\rangle} P \mid P \;{}^{A}\!\langle\!\|^{B}\; P \mid P \;{}^{A}\|\!\rangle^{B}\; P \mid & \text{parallel composition} \\
& P \longleftarrow P \mid P \longrightarrow P \mid P \underset{A}{\langle\!\|} P \mid P \underset{A}{\|\!\rangle} P \mid & \text{interleaving} \\
& P \;^{t}\; P \mid P \,_{t}\, P \mid P \underset{a}{\nabla} P \mid & \text{transfer operators} \\
& P \setminus A \mid f(P) \mid & \text{abstraction and renaming} \\
& \mu X \quad P \mid \mu X \quad P \mid \langle X_i = P_i \rangle_j & \text{recursion}
\end{array}
$$

This is the same as the syntax for the biased language, except the nondeterministic choice operators have been removed.

We define the space $\mathcal{M}_{DTB}$ (the Deterministic Model using Timed, Biased behaviours) to be those sets A of type $\mathcal{S}_{TB}$ satisfying the following healthiness conditions:

D1. $\forall \tau \quad 0 \quad \exists\, n(\tau) \quad (\tau, \sqsubseteq, s) \in A \Rightarrow \#s \quad n(\tau)$

D2. $\forall \tau \quad 0 \quad \exists\, n(\tau) \quad (\tau, \sqsubseteq, s) \in A \Rightarrow \exists\, k \quad n(\tau)\,; I_0, \ldots, I_{k-1} \in TINT$
$I_0, \ldots I_{k-1}$ partition $[0, \tau]$
$\wedge \forall i : 0 \,..\, k-1\,; t, t' \in I_i\,; \chi, \psi \in \text{bag}\,\Sigma \quad (t, \chi) \sqsubseteq (t, \psi) \Leftrightarrow (t', \chi) \sqsubseteq (t', \psi)$

D3. $(\tau, \sqsubseteq, s) \in A \wedge (t, \chi) \in \text{items} \sqsubseteq \;\Rightarrow (t, \sqsubseteq \quad t, s \quad t \quad (t, \chi)) \in A$

D4. $\forall\, \Omega : OFFREL \rightarrow EOFF \quad \exists_1 \sqsubseteq \quad (end\,\Omega, \sqsubseteq, \sqcup_{\sqsubseteq}\Omega(\sqsubseteq)) \in A$

Axioms D1–D3 are the same as axioms B1–B3 for the Prioritized Model. The fourth axiom says that the process is deterministic: given the way the environmental behaves, there is a unique offer relation that it can have; it will perform those members of the environmental offer that are maximal with respect to this offer relation.

We define a semantic function $\mathcal{A}_{DT} : DTCSP \rightarrow ENV \rightarrow \mathcal{M}_{DTB}$ such $\mathcal{A}_{DT}\ P\ \rho$ is the set of behaviours that P can perform in variable binding ρ. The semantic definitions for the constructs of the language are the same as in the Prioritized Model, except references to $\mathcal{A}_{BT}$ should be changed to $\mathcal{A}_{DT}$; for example

$$
\begin{aligned}
\mathcal{A}_{DT}\ P \;{}^{X}\!\langle\!\|^{Y}\; Q\ \rho \mathrel{\hat=} \{(\tau, \sqsubseteq_P {}^{X}\!\langle\!\|^{Y} \sqsubseteq_Q, s) \mid \Sigma s \subseteq X \cup Y \wedge (\tau, \sqsubseteq_P, s \quad X) \in \mathcal{A}_{DT}\ P\ \rho \\
\wedge (\tau, \sqsubseteq_Q, s \quad Y) \in \mathcal{A}_{DT}\ Q\ \rho\}
\end{aligned}
$$

In [Low91b] we showed that the semantic definitions respect the healthiness conditions of the semantic space. In particular, from condition D4 we see that this language is completely deterministic.

The Deterministic Model lies inside the Prioritized Model in the sense that any set of behaviours that satisfies the axioms of $\mathcal{M}_{DTB}$ also satisfies the axioms of $\mathcal{M}_{TB}$. To see this let $A \in \mathcal{M}_{DTB}$. Then A must satisfy the first three axioms of $\mathcal{M}_{TB}$ because these are the same as the first three axioms of $\mathcal{M}_{DTB}$. Taking $\Omega = \{(0, \{\!|\,|\!\})\}$ in axiom D4 we see that $\exists \sqsubseteq \quad (0, \sqsubseteq, \prec\succ) \in A$ so axiom B4 is satisfied. For axiom B5, let $(\tau, \sqsubseteq, s) \in A$, $\tau' > \tau$, $\Omega \in EOFF$ such that $I\Omega = (\tau, \tau']$. Let $\Omega' \mathrel{\widehat{=}} \{(t, s(t)) \mid 0 \quad t \quad \tau\} \cup \Omega$. Then by axiom D4 there is a unique offer relation $\sqsubseteq'$ such that $(\tau', \sqsubseteq', \sqcup_{\sqsubseteq'} \Omega') \in A$. By axiom D3 and the uniqueness condition we have $\sqsubseteq' \quad \tau = \sqsubseteq$, and by the definition of Ω' we have $\sqcup_{\sqsubseteq'} \Omega' = s \quad \sqcup_{\sqsubseteq' \upharpoonright \tau} \Omega$, so $(\tau', \sqsubseteq', s \quad \sqcup_{\sqsubseteq' \upharpoonright \tau} \Omega) \in A$ as required.

All the algebraic laws that hold in the Prioritized Model also hold in the Deterministic Model, except of course those laws relating to the uondeterministic choice operator. We also have that the external choice operators are idempotent: $P \square P = P$ and $P \square P = P$; this is a consequence of the language being completely deterministic, so couuter-examples such as the one in 3.4.5 do not occur.

Chapter 4

The Probabilistic Model

4.1 Syntax for the probabilistic language

We will now discuss the probabilistic language and model. The syntax is the same as the syntax of the biased language, except the nondeterministic choice operators are replaced by probabilistic internal choice operators, and we add a probabilistic external choice operator. The process $P\ {}_p\sqcap_q\ Q$ will act like P with probability p and like Q with probability q. The process $\ {}_{i\in I}[p_i]P_i$ will act like process P_i with probability p_i. The process $P\ {}_p\ \ {}_q\ Q$ will be biased in favour of P with probability p and biased in favour of Q with probability q. In the biased model, all nondeterminism was caused by the nondeterministic choice operators; hence the only place where nondeterminism arises in the probabilistic language is through the use of the probabilistic operators.

The complete syntax is

$$
\begin{array}{lll}
P ::= & STOP \mid SKIP \mid WAIT\ t \mid X \mid & \text{basic processes} \\
 & a \xrightarrow{t} P \mid P\quad P \mid WAIT\ t\,; P \mid & \text{sequential composition} \\
 & P\ {}_p\sqcap_q\ P \mid \quad {}_{i\in I}[p_i]P_i \mid & \text{probabilistic internal choice} \\
 & P \,\square\, P \mid P \,\square\, P \mid P\ {}_p\ \ {}_q\ P \mid c?d : D \xrightarrow{t_d} P_d \mid & \text{external choice} \\
 & P \Vert P \mid P \Vert P \mid P\ {}^A\Vert^B\ P \mid P\ {}^A\Vert^B\ P \mid & \text{parallel composition} \\
 & P \longleftarrow P \mid P \longrightarrow P \mid P \underset{A}{\Vert} P \mid P \underset{A}{\Vert} P \mid & \text{interleaving} \\
 & P\ ^t\ P \mid P\ {}_t\ P \mid P \underset{a}{\nabla} P \mid & \text{transfer operators} \\
 & P \setminus A \mid f(P) \mid & \text{abstraction and renaming} \\
 & \mu X\quad P \mid \mu X\quad P \mid \langle X_i = P_i \rangle_j & \text{recursion}
\end{array}
$$

where t and t_d range over the set $TIME$ of times, which we take to be positive real numbers; X ranges over the space VAR of variables; and a ranges over some alphabet Σ of events. I ranges over indexing sets, and is ranged over by i and j. p, q and p_i (for $i \in I$) range over the interval $(0, 1)$, with the properties that $p + q = 1$ and $\sum_{i\in I} p_I = 1$. c ranges over channel names; D is the type of the data passed on c and is ranged over by d. A and B range over Σ; f ranges over $\Sigma \to \Sigma$.

4.2 The semantic model

As before we define a behaviour or an observation of a process to be a triple $(\tau, \sqsubseteq, s)$, where

- τ is the time up until which the process is observed.
- $\sqsubseteq$ is a partial order on the space OFF ($= TIME \times \text{bag}\,\Sigma$) of offers. We say a process offers χ stronger than ψ at time t, and write $(t, \psi) \sqsubseteq (t, \chi)$, if the process gives a higher priority to the bag of events χ than the bag of events ψ at time t.
- s is a timed trace, of type $TIME \twoheadrightarrow \text{bag}\,\Sigma$: $s(t)$ is the bag of events performed at time t.

Recall the definition of the space S_{TB} of sets of prioritized behaviours:

$$S_{TB} \mathrel{\hat=} \ (BEH)$$

We also want to be able to discuss the space $\mathcal{PF}_{TB}$ (Probability Functions on Timed, Biased behaviours) of probability functions:

$$\mathcal{PF}_{TB} \mathrel{\hat=} BEH \to [0, 1]$$

We will often need to sum probabilities. We will write $\sum\{\!|f(x) \mid p(x)|\!\}$ to represent the sum of the $f(x)$, where the sum is taken over all x such that $p(x)$ holds.

We will represent a process by a pair (A, f). As before $A \in S_{TB}$ gives the set of behaviours that a process can perform. $f \in \mathcal{PF}_{TB}$ is a probability function: $f(\tau, \sqsubseteq, s)$ is the probability of $(\tau, \sqsubseteq, s)$ occurring, given a suitable environment, i.e. any environment Ω such that $(\tau, \sqsubseteq, s)$ is compatible with Ω (in the sense of section 3.3.4). We define the space $\mathcal{PP}_{TB}$ (Probabilistic Pairs using Timed Biased Behaviours) to be all such pairs:

$$\mathcal{PP}_{TB} \mathrel{\hat=} S_{TB} \times \mathcal{PF}_{TB}$$

Note that if $(\tau, \sqsubseteq, s)$ is compatible with two different environments, Ω and Ω', then the probability of $(\tau, \sqsubseteq, s)$ occurring is the same in environment Ω as in environment Ω'. This is because to say that $(\tau, \sqsubseteq, s)$ is compatible with Ω and Ω' means that both environments offer everything performed in trace s, but neither offers anything that is offered stronger under the offer relation $\sqsubseteq$: the rest of the environmental offers do not have any effect on the behaviour of the process, so the probability of $(\tau, \sqsubseteq, s)$ is the same in each environment.

It is worth stressing again the relationship between the probability function f and the environment Ω. $f(\tau, \sqsubseteq, s)$ is the probability of the process performing $(\tau, \sqsubseteq, s)$ given that $(\tau, \sqsubseteq, s)$ is compatible with Ω. We can use this to define a probability function f_Ω (for each environment Ω) which gives the probabilities of each behaviour, given that the environment offers Ω.

$$f_\Omega(\tau, \sqsubseteq, s) \mathrel{\hat=} \begin{cases} f(\tau, \sqsubseteq, s) & \text{if } (\tau, \sqsubseteq, s) \textit{ compat } \Omega \\ 0 & \text{otherwise} \end{cases}$$

In the next section we illustrate this with an example; in the following section we will formally define our semantic space.

4.2.1 Example

We present a process that models a biased coin being tossed once:

$$COIN \mathrel{\widehat{=}} head \longrightarrow STOP \;{}_{1/3}\sqcap_{2/3}\; tail \longrightarrow STOP$$

Here is a list of some of the possible behaviours of $COIN$ when it is observed up until time 2:

$$\begin{array}{lclll}
b_1 & \mathrel{\widehat{=}} & (2, [0,2] \otimes \langle \{\!|head|\!\}, \{\!|\,|\!\} \rangle & & , \prec\succ \;) \\
b_2 & \mathrel{\widehat{=}} & (2, [0,2] \otimes \langle \{\!|tail|\!\}, \{\!|\,|\!\} \rangle & & , \prec\succ \;) \\
b_3 & \mathrel{\widehat{=}} & (2, [0,1] \otimes \langle \{\!|head|\!\}, \{\!|\,|\!\} \rangle & (1,2] \otimes \langle \{\!|\,|\!\} \rangle & , \prec(1, head)\succ) \\
b_4 & \mathrel{\widehat{=}} & (2, [0,1] \otimes \langle \{\!|tail|\!\}, \{\!|\,|\!\} \rangle & (1,2] \otimes \langle \{\!|\,|\!\} \rangle & , \prec(1, tail)\succ \;)
\end{array}$$

In behaviour b_1 the probabilistic choice is made in favour of the *head*, so a *head* is offered, but nothing is performed: this must correspond to an environment where no head is offered. Behaviour b_2 is similar, except the choice is made in favour of the *tail*. In behaviour b_3 the choice is made in favour of the *head*, which is performed at time 1: this must correspond to an environment where a head is first offered at time 1. In behaviour b_4 the choice is made in favour of the *tail*, which is performed at time 1.

The probability function f associated with this process associates the following probabilities to these behaviours:

$$f(b_1) = 1/3 \qquad f(b_2) = 2/3 \qquad f(b_3) = 1/3 \qquad f(b_4) = 2/3$$

The two behaviours where the probabilistic choice is made in favour of the *head* are given probability $1/3$, while the behaviours where the choice is made in favour of the *tail* are given probability $2/3$.

Consider now an environment Ω with duration $[0,2]$ where neither a *head* nor a *tail* is offered. The behaviours b_1 and b_2 are compatible with this environment, but behaviours b_3 and b_4 are not since in both of these an event is performed that was not offered by the environment. In fact b_1 and b_2 are the *only* behaviours that $COIN$ can perform in this environment. The probability function associated with this environment has

$$f_\Omega(b_1) = 1/3 \qquad f_\Omega(b_2) = 2/3 \qquad f_\Omega(b_3) = 0 \qquad f_\Omega(b_4) = 0$$

and all other hehaviours are given probability zero. Note that the sum of the probabilities is one.

Consider now an environment Ω that first offers a head at time 1, and does not offer a tail. Now behaviours b_2 and b_3 are the possible behaviours. Behaviour b_1 is incompatihle with Ω because at time 1 it offers a *head* stronger than the empty bag, but performs the empty bag despite the fact that the environment is willing to perform a *head*: it disobeys the rule that says that at each instant the process must perform the member of the environmental offer that it offers strongest (i.e. is maximal in tbe process's offer relation). The probability function associated with this environmental offer therefore has

$$f_\Omega(b_1) = 0 \qquad f_\Omega(b_2) = 2/3 \qquad f_\Omega(b_3) = 1/3 \qquad f_\Omega(b_4) = 0$$

Finally, consider an environment that offers a *head* and a *tail* at time 1, but offers neither earlier. In this case behaviours b_3 and b_4 are possible; the other two are incompatible with the environmental offer. Hence the probability function associated with this environmental offer has

$$f_\Omega(b_1) = 0 \qquad f_\Omega(b_2) = 0 \qquad f_\Omega(b_3) = 1/3 \qquad f_\Omega(b_4) = 2/3$$

Note that the choice of whether the process offers a *head* or a *tail* is made at time 0, before either is actually offered by the environment.

4.2.2 The semantic space $\mathcal{M}_{PTB}$

We define the space $\mathcal{M}_{PTB}$ (the Probabilistic Model using Timed Biased behaviours) to be those pairs (A, f) in $\mathcal{PP}_{TB}$ satisfying the following axioms:

P1. $\forall \tau \quad 0 \quad \exists n(\tau) \quad (\tau, \sqsubseteq, s) \in A \Rightarrow \#s \quad n(\tau)$

P2. $\forall \tau \quad 0 \quad \exists n(\tau) \quad (\tau, \sqsubseteq, s) \in A \Rightarrow \exists k \quad n(\tau)\, ; I_0, \ldots, I_{k-1} \in TINT$
$I_0, \ldots I_{k-1}$ partition $[0, \tau]$
$\wedge \forall i : 0 \,..\, k - 1\, ; t, t' \in I_i\, ; \chi, \psi \in \text{bag}\,\Sigma \quad (t, \chi) \sqsubseteq (t, \psi) \Leftrightarrow (t', \chi) \sqsubseteq (t', \psi)$

P3. $(\tau, \sqsubseteq, s) \in A \wedge (t, \chi) \in \text{items} \sqsubseteq \Rightarrow (t, \sqsubseteq \quad t, s \quad t \quad (t, \chi)) \in A$

P4. $f(\tau, \sqsubseteq, s) > 0 \Leftrightarrow (\tau, \sqsubseteq, s) \in A$

P5. $\sum \{\!\!\{ f(0, \sqsubseteq, \prec\succ) \mid \sqsubseteq \in OFFREL \}\!\!\} = 1$

P6. $\forall s : TT\, ; \sqsubseteq : OFFREL\, ; \Omega : EOFF\, ; \tau, \tau' : TIME \mid \text{dom}\, s = [0, \tau] \wedge I\Omega = [\tau, \tau']$
$f(\tau, \sqsubseteq, s) = \sum \{\!\!\{ f(\tau', \sqsubseteq', s \quad \tau \quad \sqcup_{\sqsubseteq'} \Omega) \mid \sqsubseteq' \quad \tau = \sqsubseteq \}\!\!\}$

The first three of these axioms are the same as the first three axioms in the Prioritized Model. We discuss the other three axioms in turn:

P4. If the probability of a process having a certain behaviour is non-zero, then that behaviour is possible.

P5. If the environment offers no events at time 0, then the empty trace occurs with probability one.

P6. The probability of a process displaying some behaviour up to time τ is the same as the sum of the probabilities of the extensions of this behaviour that could have resulted from the environment offering Ω between times τ and τ'.

It is worth noting that in any environment there is a countable number of behaviours that a process can perform: this is a result of the syntax we have chosen, which only allows *countable* probabilistic choice. This fact means that summing over probabilities (rather than integrating) is a valid technique.

4.2.3 Laws

The following law, which was proved in section 3.3 for the Prioritized Model, also hold in this model. If a process can have a particular behaviour, then it can perform any prefix of that behaviour.

Theorem 4.2.1: $(\tau, \sqsubseteq, s) \in A \wedge \tau' \quad \tau \Rightarrow (\tau', \sqsubseteq \quad \tau', s \quad \tau') \in A.$ ♡

In addition, the following law holds in this model. If the environment offers Ω, then the sum of the probabilities of all possible behaviours is one.

Theorem 4.2.2: $\forall \Omega : EOFF \quad \forall \tau \quad end\,\Omega \quad \sum\{\!|f(end\,\Omega, \sqsubseteq, \sqcup_{\sqsubseteq}\Omega) \mid \sqsubseteq \in OFFREL|\!\} = 1.$ ♡

Proof: Pick Ω and let $\tau \mathrel{\widehat{=}} end\,\Omega$. We have

$$\begin{aligned}
& \textstyle\sum\{\!|f(end\,\Omega, \sqsubseteq, \sqcup_{\sqsubseteq}\Omega) \mid \sqsubseteq \in OFFREL|\!\} \\
= {} & \langle \text{rearranging; taking } \sqsubseteq' = \sqsubseteq \quad 0 \rangle \\
& \textstyle\sum\{\!|\sum\{\!|f(\tau, \sqsubseteq'', \sqcup_{\sqsubseteq''}\Omega) \mid \sqsubseteq'' \quad 0 = \sqsubseteq'|\!\} \mid end \sqsubseteq' = 0|\!\} \\
= {} & \langle \text{taking } s = \prec\succ \text{ in axiom P6} \rangle \\
& \textstyle\sum\{\!|f(0, \sqsubseteq', \prec\succ) \mid end \sqsubseteq' = 0|\!\} \\
= {} & \langle \text{axiom P5} \rangle \\
& 1
\end{aligned}$$

□

4.2.4 Semantic functions

We define the space of variable bindings for the Probabilistic Model by

$$ENV_P \mathrel{\widehat{=}} VAR \rightarrow \mathcal{PP}_{TB}$$

We shall drop the subscript P where it is obvious from the context which model we are working in. In the next section we shall define functions $\mathcal{A}_{PBT} : PBTCSP \rightarrow ENV_P \rightarrow \mathcal{S}_{TB}$ and $\mathcal{P}_{PBT} : PBTCSP \rightarrow ENV_P \rightarrow \mathcal{PF}_{TB}$ such that in variable binding ρ, $\mathcal{A}_{PBT}\ P\ \rho$ gives the set of possible behaviours of process P and $\mathcal{P}_{PBT}\ P\ \rho$ gives the behaviour probability function. We define the semantic function $\mathcal{F}_{PBT} : PBTCSP \rightarrow ENV_P \rightarrow \mathcal{M}_{PTB}$ by $\mathcal{F}_{PBT}\ P\ \rho \mathrel{\widehat{=}} (\mathcal{A}_{PBT}\ P\ \rho, \mathcal{P}_{PBT}\ P\ \rho)$. In section 4.3.8 we discuss which algebraic laws hold in this model. The semantic definitions were proved sound in [Low91b].

4.3 Semantic definitions

In this section we derive the semantic definitions for each of our basic processes and for each of our operators. For most of the processes (all except the probabilistic operators and recursion) the definition of the set A of possible behaviours is the same as in the biased model; for these processes we derive the definitions for the probability functions from the definition of A. For the probabilistic operators, the definitions are easy; for recursion, the definitions are very similar to those in the biased model.

The definitions are summarized in appendix A.

4.3.1 Basic processes

The processes $STOP$, $SKIP$ and $WAIT\ t$ are completely deterministic. The semantic definitions for their sets of possible behaviours are the same as in the biased model. Each of these semantic definitions are of the form

$$\mathcal{A}_{PBT}\ P\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \mid S(\tau, \sqsubseteq, s)\}$$

for some predicate S. Behaviours of this form occur with probability one; all other behaviours have probability zero. This gives the following definition:

$$\mathcal{P}_{PBT}\ P\ \rho \mathrel{\widehat{=}} \text{fillout}\{(\tau, \sqsubseteq, s) \mapsto 1 \mid S(\tau, \sqsubseteq, s)\}$$

where the function fillout : $(BEH \nrightarrow [0,1]) \to (BEH \to [0,1])$ extends partial behaviour probability functions to total probability functions:

$$\forall f, (\tau, \sqsubseteq, s) \quad \text{fillout}\ f\ (\tau, \sqsubseteq, s) = \begin{cases} f(\tau, \sqsubseteq, s) & (\tau, \sqsubseteq, s) \in \operatorname{dom} f \\ 0 & (\tau, \sqsubseteq, s) \notin \operatorname{dom} f \end{cases}$$

All behaviours not defined in f are assumed not to occur, and so are given zero probability.

4.3.2 Unary operators

Let F be one of the unary operators prefixing, hiding, renaming, or delay. For each of these operators, the definition of the set of possible behaviours from section 3.4 carry over to the Probabilistic Model. In each case the semantic definition can be put into the form

$$\mathcal{A}_{PBT}\ F(P)\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \mid \exists \tau', \sqsubseteq', s' \quad (\tau', \sqsubseteq', s') \in \mathcal{A}_{PBT}\ P\ \rho \wedge S(\tau, \tau', \sqsubseteq, \sqsubseteq', s, s')\}$$

for some predicate S. The probability of $F(P)$ performing a behaviour $(\tau, \sqsubseteq, s)$ is the probability of P performing some corresponding behaviour $(\tau', \sqsubseteq', s')$ such that $S(\tau, \tau', \sqsubseteq, \sqsubseteq', s, s')$ holds; hence we want to sum over all such behaviours. This gives the following definition

$$\mathcal{P}_{PBT}\ F(P)\ \rho(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} \sum \{\!\!\{ \mathcal{P}_{PBT}\ P\ \rho(\tau', \sqsubseteq', s') \mid S(\tau, \tau', \sqsubseteq, \sqsubseteq', s, s') \}\!\!\}$$

Note that this can normally be greatly simplified using the one-point rule.

4.3.3 Binary operators

Let $\_\oplus\_$ be one of the binary operators on the syntax, other than the probabilistic operators. Again, for each of these operators the semantic definition from section 3.4 carries over to the Probabilistic Model. The definition for the set of possible behaviours can be put in the form

$$\begin{aligned} \mathcal{A}_{PBT}\ P \oplus Q\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \mid \exists \tau_P, \tau_Q, \sqsubseteq_P, \sqsubseteq_Q, s_P, s_Q \\ &(\tau_P, \sqsubseteq_P, s_P) \in \mathcal{A}_{PBT}\ P\ \rho \wedge (\tau_Q, \sqsubseteq_Q, s_Q) \in \mathcal{A}_{PBT}\ Q\ \rho \\ &\wedge S(\tau, \tau_P, \tau_Q, \sqsubseteq, \sqsubseteq_P, \sqsubseteq_Q, s, s_P, s_Q)\} \end{aligned}$$

for some predicate S. The probability of $P \oplus Q$ performing such a behaviour $(\tau, \sqsubseteq, s)$ is the probability of P and Q performing some corresponding behaviours $(\tau_P, \sqsubseteq_P, s_P)$ and $(\tau_Q, \sqsubseteq_Q, s_Q)$ such that $S(\tau, \tau_P, \tau_Q, \sqsubseteq, \sqsubseteq_P, \sqsubseteq_Q, s, s_P, s_Q)$ holds; hence we want to sum over all such behaviours. This gives the following definition:

$$\begin{aligned}
&\mathcal{P}_{PBT}\ P \oplus Q\ \rho(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} \\
&\quad \sum \{\!|\, \mathcal{P}_{PBT}\ P\ \rho(\tau_P, \sqsubseteq_P, s_P) \times \mathcal{P}_{PBT}\ Q\ \rho(\tau_Q, \sqsubseteq_Q, s_Q) \mid S(\tau, \tau_P, \tau_Q, \sqsubseteq, \sqsubseteq_P, \sqsubseteq_Q, s, s_P, s_Q) \,|\!\}
\end{aligned}$$

4.3.4 Communication

The definition for the set of possible behaviours for the prefix choice operator is

$$\begin{aligned}
\mathcal{A}_{PBT}\ c?a : A \xrightarrow{0} P_a\ \rho \mathrel{\widehat{=}} &\{(\tau, [0, \tau] \otimes \langle \{\!| c?a |\!\}, \{\!||\!\} \rangle, \prec\succ) \mid \tau \in \mathit{TIME}\} \\
&\cup \{(\tau, [0, t] \otimes \langle \{\!| c?a |\!\}, \{\!||\!\} \rangle \quad \sqsubseteq + t, (t, c?\hat{a}) \quad s + t) \mid \\
&\qquad \hat{a} \in A \wedge t \quad \tau \wedge (\tau - t, \{0\} \otimes \langle \{\!||\!\} \rangle \quad \sqsubseteq, s) \in \mathcal{A}_{PBT}\ P_{\hat{a}}\ \rho\}
\end{aligned}$$

For the probability function, behaviours of the first sort occur with probability one, if the environment is unwilling to communicate on c. The probability of a behaviour of the second sort is the probability of $P_{\hat{a}}$ performing the corresponding behaviour starting at time t when the first communication the environment is willing to make is an $\hat{a}$ at time t.

$$\begin{aligned}
&\mathcal{P}_{PBT}\ c?a : A \xrightarrow{0} P_a\ \rho \mathrel{\widehat{=}} \\
&\quad \text{fillout} \begin{pmatrix}
\{(\tau, [0, \tau] \otimes \langle \{\!| c?a |\!\}, \{\!||\!\} \rangle, \prec\succ) \mapsto 1 \mid \tau \in \mathit{TIME}\} \\
\cup \{(\tau, [0, t] \otimes \langle \{\!| c?a |\!\}, \{\!||\!\} \rangle \quad \sqsubseteq + t, (t, c?\hat{a}) \quad s + t) \mapsto \\
\mathcal{P}_{PBT}\ P_{\hat{a}}\ \rho(\tau - t, \{0\} \otimes \langle \{\!||\!\} \rangle \quad \sqsubseteq, s) \mid \\
\hat{a} \in A \wedge t \quad \tau\}
\end{pmatrix}
\end{aligned}$$

As in the Prioritized Model, we can use this to define the general prefix choice operator:

$$c?a : A \xrightarrow{t_a} P_a \mathrel{\widehat{=}} c?a : A \xrightarrow{0} (\mathit{WAIT}\ t_a\ ; P_a)$$

4.3.5 Probabilistic internal choice

The process $P\ {}_p\sqcap_q\ Q$ acts like P with probability p, and like Q with probability q. It will have behaviour $(\tau, \sqsubseteq, s)$ if

- P is chosen and P has behaviour $(\tau, \sqsubseteq, s)$,
- or Q is chosen and Q has behaviour $(\tau, \sqsubseteq, s)$.

We therefore have the following definitions, assuming $p \neq 0$, $q \neq 0$, and $p + q = 1$:

$$\begin{aligned}
\mathcal{A}_{PBT}\ P\ {}_p\sqcap_q\ Q\ \rho &\mathrel{\widehat{=}} \mathcal{A}_{PBT}\ P\ \rho \cup \mathcal{A}_{PBT}\ Q\ \rho \\
\mathcal{P}_{PBT}\ P\ {}_p\sqcap_q\ Q\ \rho(\tau, \sqsubseteq, s) &\mathrel{\widehat{=}} p.\mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) + q.\mathcal{P}_{PBT}\ Q\ \rho(\tau, \sqsubseteq, s)
\end{aligned}$$

Infinite probabilistic choice

If I is a countable set, and $\sum_{i \in I} p_i = 1$ then we will write $_{i \in I}[p_i]P_i$ to represent the process that, for all i, acts like process P_i with probability p_i.

We give a semantics to this process in the obvious way:

$$\mathcal{A}_{PBT} \quad _{i \in I}[p_i]P_i \ \rho \ \widehat{=} \ \bigcup\{\mathcal{A}_{PBT} \ P_i \ \rho \mid i \in I\}$$
$$\mathcal{P}_{PBT} \quad _{i \in I}[p_i]P_i \ \rho(\tau, \sqsubseteq, s) \ \widehat{=} \ \sum\{\!\!\{p_i \times \mathcal{P}_{PBT} \ P_i \ \rho(\tau, \sqsubseteq, s) \mid i \in I\}\!\!\}$$

This is well defined only when the set of processes $\{P_i \mid i \in I\}$ is uniformly bounded in the sense of section 3.4.10.

As in the Prioritized Model, this method does not always effectively model nondeterminism that does not manifest itself in a finite amount of time. For example, consider the process P which can perform any finite number of as:

$$P \ \widehat{=} \quad _{n \in \mathbb{N}}[(1/2)^{n+1}]P_n \qquad \text{where} \qquad P_0 \ \widehat{=} \ STOP \qquad P_{n+1} \ \widehat{=} \ a \xrightarrow{1} P_n$$

We would expect this to be different from the process P' that can perform an arbitrary number of as:

$$P' \ \widehat{=} \ a \xrightarrow{1} P' \ _{1/2}\sqcap_{1/2} \ STOP$$

However, our semantics gives the same value to both of these processes. It is interesting that the behaviours of P' that our model does not adequately represent — namely where an infinite number of as are performed — occur with zero probability.

4.3.6 Probabilistic external choice

In this section we describe a probabilistic external choice operator $_p \quad _q$ such that $P \ _p \quad _q \ Q$ offers an external choice between P and Q that is biased in favour of P with probability p, and biased in favour of Q with probability q. The probabilistic external choice operator is defined by

$$P \ _p \quad _q \ Q \ \widehat{=} \ (P \sqsubset\!\!\!\Box \ Q) \ _p\sqcap_q \ (P \ \Box\!\!\!\sqsupset \ Q)$$

$P \ _p \quad _q \ Q$ acts like $P \sqsubset\!\!\!\Box \ Q$ with probability p, and like $P \ \Box\!\!\!\sqsupset \ Q$ with probability q.

This operator is very similar to the probabilistic choice operator defined in most probabilistic models of CCS, for example in [vGSST90]. There, an external choice between processes P and Q is written $[p]P + [q]Q$: if the environment can perform events of both P and Q then the choice is made in favour of P with probability p and in favour of Q with probability q. This can then be used to define a probabilistic internal choice between two processes by $[p]\tau.P + [q]\tau.Q$, where τ represents an internal action. Our approach has been the other way round: we have defined biased external choice operators and a probabilistic internal choice operator, and used these to define a probabilistic external choice operator. We believe that it is more natural to define separate internal and external choice operators since these are very different operations. A language with more operators, while being harder to reason *about*, is easier to reason *with*.

4.3.7 Recursion

Our definition of recursion for probabilistic processes follows closely our approach for prioritized processes. We define a metric on the space $\mathcal{M}_{PTB}$ by considering the first time at which two processes may be distinguished. We define an operator on behaviour sets and behaviour probability functions which gives the behaviour of a process up to a certain time.

$$A \quad t \mathrel{\hat=} \{(\tau, \sqsubseteq, s) \in A \mid \tau \quad t\} \qquad f \quad t \mathrel{\hat=} \{(\tau, \sqsubseteq, s) \mapsto f(\tau, \sqsubseteq, s) \mid \tau \quad t\}$$

We define the metric on $\mathcal{M}_{PTB}$ by

$$d((A_P, f_P), (A_Q, f_Q)) \mathrel{\hat=} \inf(\{2^{-t} \mid A_P \quad t = A_Q \quad t \wedge f_P \quad t = f_Q \quad t\} \cup \{1\})$$

We define the mapping on the semantic space corresponding to a term:

$$M(X, P)\rho \mathrel{\hat=} \lambda\, Y \quad \mathcal{F}_{PBT} \ P \ \rho[Y/X]$$

We can then define recursion by

$$\mathcal{F}_{PBT} \ \mu\, X \quad P \ \mathrel{\hat=} \text{the unique fixed point of } M(X, P)\rho$$

As in the Prioritized Model, this is well defined when P is constructive for X.

Delayed recursion

For delayed recursion, we define a mapping W_δ which delays it argument by δ:

$$W_\delta \mathrel{\hat=} \lambda\, Y \quad \mathcal{F}_{PBT} \ \ WAIT\ \delta\,;\, X \ \ \rho[Y/X]$$

We can now define delayed recursion by

$$\mathcal{F}_{PBT} \ \mu\, X \quad P \ \rho \mathrel{\hat=} \text{the unique fixed point of } M(X, P)\rho \circ W_\delta$$

Mutual recursion

In order to give semantics to $\langle X_i = P_i \mid i \in I\rangle_j$ we consider the semantic domain $\mathcal{PP}^I_{TB}$, i.e. the product space with one copy of $\mathcal{PP}_{TB}$ for each element of I. We define a metric on this space by

$$\underline{d}(\underline{U}, \underline{V}) \mathrel{\hat=} \sup\{d(U_i, V_i) \mid i \in I\}$$

If $\underline{P}$ is a vector of PBTCSP terms, $\underline{X}$ is a vector of variables, and $\underline{Y}$ a vector of members of $\mathcal{PP}_{TB}$, all indexed by set I, then the mapping on $\mathcal{PP}^I_{TB}$ corresponding to $\underline{P}$ is given by

$$M(\underline{X}, \underline{P})\rho \mathrel{\hat=} \lambda\, \underline{Y} \quad \mathcal{F}_{PBT} \ \underline{P} \ \rho[Y_i/X_i \mid i \in I]$$

We can now give a semantics to mutual recursion. If $\underline{P}$ is a vector of PBTCSP terms, then

$$\mathcal{F}_{PBT} \ \langle X_i = P_i \mid i \in I\rangle_j \ \rho \mathrel{\hat=} S_j \text{ where } \underline{S} \text{ is a fixed point of } M(\underline{X}, \underline{P})\rho$$

As in the Prioritized Model, this is well defined when the vector of terms P is constructive for vector of variables X. The proof of this appeared in [Low92a] and is very similar to the proof sketched in chapter 3. We defined a second vector of terms $\underline{Q}$ by

$$Q_i \mathrel{\hat=} P_i[Q_j/X_j \mid j \in seg(i)]$$

we showed that this vector is well defined; we showed that the corresponding mapping $M(\underline{X}, \underline{Q})\rho$ is a contraction mapping and so has a unique fixed point; we showed that this fixed point is also a fixed point of $M(\underline{X}, \underline{P})\rho$; we showed that this fixed point is unique.

4.3.8 Algebraic laws

In this section we discuss which algebraic laws hold in the probabilistic language. The proofs of these laws are similar to the proofs for the Prioritized Model.

All the laws that were described above for the Prioritized Model carry forward to this model (except of course those laws involving the nondeterministic choice operator). In addition, the following laws hold for the probabilistic choice operator:

Law 4.3.1 (Commutativity of probabilistic choice) $P \;{}_p\sqcap_q\; Q = Q \;{}_q\sqcap_p\; P.$ $\triangle$

Law 4.3.2 (Idempotence of probabilistic choice) $P \;{}_p\sqcap_q\; P = P.$ $\triangle$

Law 4.3.3 (Associativity of probabilistic choice)

$$P \;{}_p\sqcap_{q+r}\; (Q \;{}_{q/q+r}\sqcap_{r/q+r}\; R) = (P \;{}_{p/p+q}\sqcap_{q/p+q}\; Q) \;{}_{p+q}\sqcap_r\; R$$

$\triangle$

Law 4.3.4 (Distributivity) All operators except recursion distribute through probabilistic choice:

Prefixing:	$a \xrightarrow{t} (P \;{}_p\sqcap_q\; Q) = a \xrightarrow{t} P \;{}_p\sqcap_q\; a \xrightarrow{t} Q$
External choice:	$P \,\square\, (Q \;{}_p\sqcap_q\; R) = P \,\square\, Q \;{}_p\sqcap_q\; P \,\square\, R$
	$(P \;{}_p\sqcap_q\; Q) \,\square\, R = P \,\square\, R \;{}_p\sqcap_q\; Q \,\square\, R$
Parallel composition:	$P \;{}^X\!\|^Y\; (Q \;{}_p\sqcap_q\; R) = P \;{}^X\!\|^Y\; Q \;{}_p\sqcap_q\; P \;{}^X\!\|^Y\; R$
	$(P \;{}_p\sqcap_q\; Q) \;{}^X\!\|^Y\; R = P \;{}^X\!\|^Y\; R \;{}_p\sqcap_q\; Q \;{}^X\!\|^Y\; R$
Interleaving:	$P \longleftarrow (Q \;{}_p\sqcap_q\; R) = P \longleftarrow Q \;{}_p\sqcap_q\; P \longleftarrow R$
	$(P \;{}_p\sqcap_q\; Q) \longleftarrow R = P \longleftarrow R \;{}_p\sqcap_q\; Q \longleftarrow R$
Hiding:	$(P \;{}_p\sqcap_q\; Q) \setminus X = P \setminus X \;{}_p\sqcap_q\; Q \setminus X$
Renaming:	$f(P \;{}_p\sqcap_q\; Q) = f(P) \;{}_p\sqcap_q\; f(Q)$
Sequential composition:	$(P \;{}_p\sqcap_q\; Q) \quad R = P \quad R \;{}_p\sqcap_q\; Q \quad R$
	$P \quad (Q \;{}_p\sqcap_q\; R) = P \quad Q \;{}_p\sqcap_q\; P \quad R$

and similar laws for the right biased operators. $\triangle$

4.4 Example: a communications protocol

We consider a very simple communications protocol transmitting over an unreliable medium. For simplicity, we abstract away from the actual contents of the communication and just concentrate on whether the message is transmitted. We also only insist that the protocol is

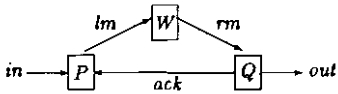

Figure 4.1: The communications protocol

able to handle a single message. We are interested in the probability of the message being correctly transmitted within a certain time.

The protocol is as in figure 4.1. Messages are received on the channel *in*. They are then passed along the wire W, which loses a proportion of its inputs. If Q receives the message, it acknowledges it on the channel *ack* and outputs on *out*. If P does not receive an acknowledgement within a certain amonnt of time, then it tries retransmitting.

The processes P, Q and W are defined by

$$\begin{aligned} P &\mathrel{\widehat=} in \longrightarrow \mu X \;\; lm \longrightarrow (ack \longrightarrow STOP \,\Box\, WAIT\ 1 - 2\delta\,;X) \\ Q &\mathrel{\widehat=} rm \longrightarrow ack \xrightarrow{1-3\delta} out \longrightarrow STOP \\ W &\mathrel{\widehat=} \mu X \;\; lm \longrightarrow ((rm \longrightarrow X)\,{}_p\sqcap_q\, X) \end{aligned}$$

The protocol is then given by

$$PROTOCOL \mathrel{\widehat=} ((P\ {}^{A}\|^{B}\ W)\ {}^{A\cup B}\|^{C}\ Q) \setminus Y$$

where A, B, and C are the alphabets of P, W, and Q, and Y is the set of internal actions:

$$A \mathrel{\widehat=} \{in, lm, ack\} \qquad B \mathrel{\widehat=} \{lm, rm\} \qquad C \mathrel{\widehat=} \{rm, out, ack\} \qquad Y \mathrel{\widehat=} \{lm, rm, ack\}$$

For simplicity we rewrite P and W by

$$\begin{aligned} P &= in \longrightarrow P_1 \\ P_1 &= lm \longrightarrow (ack \longrightarrow STOP \,\Box\, WAIT\ 1 - \delta\,; P_1) \\ W &= lm \longrightarrow (rm \xrightarrow{2\delta} W\,{}_p\sqcap_q\, WAIT\ \delta\,; W) \end{aligned}$$

Then using laws for communication and hiding we have $PROTOCOL = in \longrightarrow PROTOCOL'$ where $PROTOCOL' \mathrel{\widehat=} ((P_1\ {}^{A}\|^{B}\ W)\ {}^{A\cup B}\|^{C}\ Q) \setminus Y$. Now

$$\begin{aligned} & PROTOCOL' \\ = {} & \left\langle \text{laws of communication; parallel composition distributes through probabilistic choice} \right\rangle \\ & (lm \longrightarrow ((ack \longrightarrow STOP \,\Box\, WAIT\ 1 - \delta\,; P_1)\ {}^{A}\|^{B}\ (rm \xrightarrow{2\delta} W)) \\ & \qquad\qquad {}^{A\cup B}\|^{C} (rm \longrightarrow ack \xrightarrow{1-3\delta} ;out \longrightarrow STOP) \\ & \qquad {}_p\sqcap_q \\ & \qquad ((ack \longrightarrow STOP \,\Box\, WAIT\ 1 - \delta\,; P_1)\ {}^{A}\|^{B}\ (WAIT\ \delta\,; W)) \\ & \qquad\qquad {}^{A\cup B}\|^{C} (rm \longrightarrow ack \xrightarrow{1-3\delta} ;out \longrightarrow STOP)) \setminus Y \end{aligned}$$

$$= \langle \text{laws of communication and hidiug} \rangle$$
$$(lm \longrightarrow (rm \longrightarrow ack \xrightarrow{1-3\delta} out \longrightarrow STOP \;{}_p\sqcap_q\; WAIT\ 1\ ; (P_1 \;{}^{A}\|^{B}\; W) \;{}^{A\cup B}\|^{C}\; Q)) \setminus Y$$
$$= \langle \text{laws of biding} \rangle$$
$$WAIT\ 1 - \delta\ ; out \;{}_p\sqcap_q\; WAIT\ 1\ ; PROTOCOL'$$

Let q_n be the probability that $PROTOCOL'$ is *not* willing to perform *out* within $n - \delta$ seconds ($n \in$). Evidently $q_0 = 1$ and $q_{n+1} = q.q_n$. Hence $q_n = q^n$ and so the protocol *is* willing to perform *out* within n seconds of receiving an input with probability $1 - q^n$. Letting n tend to infinity we see that the protocol is eventually willing to perform *out* with probability one.

In chapter 7 we will study a somewhat more reasonable protocol, which is able to handle more than one message. We will prove that it acts like a one-place buffer and will present a probabilistic investigation of the time taken for messages to be transmitted.

Chapter 5

Specification and Proof of Prioritized Processes

In chapter 3 we gave a semantic model for a language using prioritized operators. Unfortunately, the semantic equations are rather complicated and so hard to use for reasoning about processes. In this chapter we present a proof system, in the style of the proof system described in section 2.4, which will enable us to prove that a process meets its specification. The proof system will comprise a number of inference rules; these rules will allow proof obligations on composite processes to be reduced to proof obligations on the subcomponents.

In section 5.1 we describe the form of our specifications. If $S(\tau, \sqsubseteq, s)$ is a predicate on behaviours, we will say that a process P satisfies $S(\tau, \sqsubseteq, s)$ in environment ρ, written P $\mathbf{sat}_\rho$ $S(\tau, \sqsubseteq, s)$, if $S(\tau, \sqsubseteq, s)$ is true for all behaviours $(\tau, \sqsubseteq, s)$ of P. We will normally drop the argument $(\tau, \sqsubseteq, s)$ of S and simply write P $\mathbf{sat}_\rho$ S when it is obvious from the context in which model we are working.

This method can be extended to the Probabilistic and Deterministic Models. In section 5.2 we will present abstraction mappings from these two models to the Prioritized Model and show that a probabilistic or deterministic process satisfies a behavioural specification if the corresponding biased process satisfies the same specification. Note that this proof system will only relate to non-probabilistic specifications, i.e. specifications that state that *all* behaviours of a process satisfy some property. In chapter 7 we will present a proof system that allows us to prove probabilistic specifications on processes, for example specifications such as 'an a is offered within 3 seconds with probability 80%'.

In section 5.3 we present a language for specifying processes. This will be based on the specification language described in section 2.5, extended so as to be able to talk about priorities.

We derive inference rules for each of the constructs of the language in section 5.4. We also show that they are complete in the sense that if, from the semantic definitions, a predicate $S(\tau, \sqsubseteq, s)$ can be shown to be true of all the behaviours of a process P, then P $\mathbf{sat}_\rho$ S can be proved using the proof system.

In section 5.5 we apply our proof system to the lift system introduced in section 3.2: we show that the lift always arrives on a particular floor within 15 seconds of being summoned.

5.1 Specification of prioritized processes

We define a behavioural specification to be a predicate $S(\tau, \sqsubseteq, s)$ with free variable representing a possible behaviour. Our basic specification statement will be of the form P $\mathbf{sat}_\rho$ $S(\tau, \sqsubseteq, s)$ **in** $\mathcal{M}_{TB}$. This will mean that in environment ρ *all* behaviours $(\tau, \sqsubseteq, s)$ of P will satisfy the predicate $S(\tau, \sqsubseteq, s)$:

Definition 5.1.1: $P\ \mathbf{sat}_\rho\ S(\tau, \sqsubseteq, s)\ \mathbf{in}\ \mathcal{M}_{TB} \mathrel{\widehat{=}} \forall (\tau, \sqsubseteq, s) \in \mathcal{A}_{PBT}\ P\ \rho \quad S(\tau, \sqsubseteq, s)$. $\diamond$

If P is a process, we may omit reference to the environment:

Definition 5.1.2:

$$P\ \mathbf{sat}\ S(\tau, \sqsubseteq, s)\ \mathbf{in}\ \mathcal{M}_{TB} \mathrel{\widehat{=}} \forall \rho \in ENV \quad \forall (\tau, \sqsubseteq, s) \in \mathcal{A}_{PBT}\ P\ \rho \quad S(\tau, \sqsubseteq, s)$$

$\diamond$

We shall omit this qualification '**in** $\mathcal{M}_{TB}$' and the argument $(\tau, \sqsubseteq, s)$ of S where the model we are working in is obvious from the context.

5.2 Abstraction mappings

In the following two subsections we give abstraction mappings from the Probabilistic and Deterministic Models to the (unprobabilistic) Prioritized Model. These abstraction results will allow us to prove results about probabilistic or deterministic processes by proving corresponding results about the corresponding process in the Biased Model. In chapter 6 we will also give an abstraction mapping from the Prioritized Model to the Timed Failures Model of Timed CSP. The relationships between the probabilistic, deterministic and prioritized languages and models are shown in figure 5.1. The mappings $\varphi_P^{(B)}$ and $\theta_P^{(B)}$ remove *p*robabilities but keep *b*iases; the mappings $\varphi_D^{(B)}$ and $\theta_D^{(B)}$ remove *d*eterminism but keep *b*iases.

$$\begin{tikzcd}
PBTCSP \arrow[dr, "\varphi_P^{(B)}"'] & & DTCSP \arrow[dl, "\varphi_D^{(B)}"] \\
& BTCSP &
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathcal{M}_{PTB} \arrow[dr, "\theta_P^{(B)}"'] & & \mathcal{M}_{DTB} \arrow[dl, "\theta_D^{(B)}"] \\
& \mathcal{M}_{TB} &
\end{tikzcd}$$

Figure 5.1: A hierarchy of languages and models

5.2.1 Abstraction from the Probabilistic Model

In this section we give an abstraction mapping from the Probabilistic Model to the Prioritized Model. We define a mapping $\varphi_P^{(B)} : PBTCSP \to BTCSP$ that removes all probabilities from the syntax: $\varphi_P^{(B)}$ maps probabilistic choices to nondeterministic choices and distributes through all other operators:

$$
\begin{aligned}
\varphi_P^{(B)}(P\ {}_p\sqcap_q\ Q) &\mathrel{\hat=} \varphi_P^{(B)}(P) \sqcap \varphi_P^{(B)}(Q) \\
\varphi_P^{(B)}(\ \ {}_{i\in I}[p_i]P_i) &\mathrel{\hat=} \ \ {}_{i\in I}\,\varphi_P^{(B)}(P_i) \\
\varphi_P^{(B)}(P\ {}_p\ \ {}_q\ Q) &\mathrel{\hat=} P' \,\square\, Q' \sqcap P' \,\square\, Q' && \text{where } P' \mathrel{\hat=} \varphi_P^{(B)}(P) \text{ and } Q' \mathrel{\hat=} \varphi_P^{(B)}(Q) \\
\varphi_P^{(B)}(P) &\mathrel{\hat=} P && \text{for } P = \mathit{STOP}, \mathit{SKIP}, \mathit{WAIT}\ t, \text{ or } X \\
\varphi_P^{(B)}(F(P)) &\mathrel{\hat=} F(\varphi(P)) && \text{for } F(P) = a \xrightarrow{t} P, \mathit{WAIT}\ t\,; P, P \setminus X, \\
&&& \quad f(P), \mu X\ \ P, \text{ or } \mu X\ \ P \\
\varphi_P^{(B)}(P \oplus Q) &\mathrel{\hat=} \varphi(P) \oplus \varphi(Q) && \text{for } \oplus = \ ,\square,\square,\Vert,\Vert, {}_X\Vert_Y, {}_X\Vert_Y, \\
&&& \quad \longleftarrow, \longrightarrow, \underset{C}{\Vert}, \underset{C}{\Vert}, \overset{t}{\ }, {}_t, \text{ or } \underset{a}{\nabla} \\
\varphi_P^{(B)}(c?a : A \xrightarrow{t_a} P_a) &\mathrel{\hat=} c?a : A \xrightarrow{t_a} \varphi_P^{(B)}(P_a) \\
\varphi_P^{(B)}(\langle X_i = P_i\rangle_j) &\mathrel{\hat=} \langle X_i = \varphi_P^{(B)}(P_i)\rangle_j
\end{aligned}
$$

The corresponding semantic map $\theta_P^{(B)} : \mathcal{M}_{PTB} \to \mathcal{M}_{TB}$ is easy to define: the process with set of possible behaviours A and probability function f maps to the process with set of possible behaviours A.

$$\theta_P^{(B)}(A,f) \mathrel{\hat=} A$$

so $\theta_P^{(B)}$ is the projection π_I. We can show that $\theta_P^{(B)}$ maps $\mathcal{M}_{PTB}$ into $\mathcal{M}_{TB}$.

Theorem 5.2.1: $\theta_P^{(B)}(\mathcal{M}_{PTB}) \subseteq \mathcal{M}_{TB}$. ♡

Proof: We must show that for all $(A,f) \in \mathcal{M}_{PTB}$, the set A satisfies the healthiness conditions of $\mathcal{M}_{TB}$. The first three conditions are easy as they are the same as the first three healthiness conditions of $\mathcal{M}_{PTB}$. For condition B4 note that by axiom P5 of $\mathcal{M}_{PTB}$ we have some offer relation $\sqsubseteq$ such that $f(0, \sqsubseteq, \langle\rangle) > 0$, so by condition P4 we have $(0, \sqsubseteq, \langle\rangle) \in A$. For condition B5, suppose $(\tau, \sqsubseteq, s) \in A$, $\tau' > \tau$ and $\Omega \in \mathit{EOFF}$ with $I\Omega = (\tau, \tau']$. Then $f(\tau, \sqsubseteq, s) > 0$ by condition P4. Let $\Omega' \mathrel{\hat=} \{(\tau, s(\tau))\} \cup \Omega$; then by condition P6

$$\sum \left\{\!\!\left| f(\tau', \sqsubseteq', s\ \ \tau\ \ \sqcup_{\sqsubseteq' \restriction \tau} \Omega') \mid \sqsubseteq'\ \ \tau = \sqsubseteq \right|\!\!\right\} > 0$$

Hence there is some offer relation $\sqsubseteq'$ such that $f(\tau', \sqsubseteq', s\ \ \tau\ \ \sqcup_{\sqsubseteq' \restriction \tau} \Omega') > 0$ and $\sqsubseteq'\ \ \tau = \sqsubseteq$. But $s\ \ \tau\ \ \sqcup_{\sqsubseteq' \restriction \tau} \Omega' = s\ \ \sqcup_{\sqsubseteq' \restriction \tau} \Omega$; so by condition P4, $(\tau', \sqsubseteq', s\ \ \sqcup_{\sqsubseteq' \restriction \tau} \Omega) \in A$, as required. □

We will now prove an abstraction theorem that says that $\theta_P^{(B)}(\mathcal{F}_{PBT}\ \ P\ \ \rho) = \mathcal{A}_{BT}\ \ \varphi_P^{(B)}(P)\ \ \rho'$ for suitable environments ρ and ρ'; the condition on the environments is that $\pi_I(\rho\ \ X\ \ ) = \rho'\ \ X$ for all variables X; this can be written as $\rho' = \pi_I \circ \rho$.

Theorem 5.2.2: If $\rho' = \pi_I \circ \rho$, then

$$\theta_P^{(B)}(\mathcal{F}_{PBT}\ \ P\ \ \rho) = \mathcal{A}_{PBT}\ \ P\ \ \rho = \mathcal{A}_{BT}\ \ \varphi_P^{(B)}(P)\ \ \rho'$$

♡

Proof: This can be proved by structural induction; all cases are easy because the semantic definitions are very similar in the two models. We prove the result for probabilistic choice as an example. Assume $\rho' = \pi_I \circ \rho$; then

$$\begin{aligned}
& \mathcal{A}_{PBT}\ P\ {}_p\sqcap_q\ Q\ \rho \\
= & \left\langle \text{semantic definition in } \mathcal{M}_{PTB} \right\rangle \\
& \mathcal{A}_{PBT}\ P\ \rho \cup \mathcal{A}_{PBT}\ Q\ \rho \\
= & \left\langle \text{inductive hypothesis} \right\rangle \\
& \mathcal{A}_{BT}\ \varphi_P^{(B)}(P)\ \rho' \cup \mathcal{A}_{BT}\ \varphi_P^{(B)}(Q)\ \rho' \\
= & \left\langle \text{semantic definition in } \mathcal{M}_{TB} \right\rangle \\
& \mathcal{A}_{BT}\ \varphi_P^{(B)}(P) \sqcap \varphi_P^{(B)}(Q)\ \rho' \\
= & \left\langle \text{definition of } \varphi_P^{(B)} \right\rangle \\
& \mathcal{A}_{BT}\ \varphi_P^{(B)}(P\ {}_p\sqcap_q\ Q)\ \rho'
\end{aligned}$$

□

If we define a satisfaction relation in $\mathcal{M}_{PTB}$ by

$$P\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{PTB} \Leftrightarrow \forall (\tau, \sqsubseteq, s) \in \mathcal{A}_{PBT}\ P \quad S(\tau, \sqsubseteq, s)$$

then we have the following inference rule

Rule 5.2.3:

$$\frac{\varphi_P^{(B)}(P)\ \mathbf{sat}_{\rho'}\ S \text{ in } \mathcal{M}_{TB}}{P\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{PTB}} \left[\rho' = \pi_I \circ \rho \right]$$

△

Proof: We have

$$\begin{aligned}
& \varphi_P^{(B)}(P)\ \mathbf{sat}_{\rho'}\ S \text{ in } \mathcal{M}_{TB} \\
\Leftrightarrow & \left\langle \text{defiuition of } \mathbf{sat} \text{ in } \mathcal{M}_{TB} \right\rangle \\
& \forall (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ \varphi_P^{(B)}(P)\ \rho' \quad S(\tau, \sqsubseteq, s) \\
\Leftrightarrow & \left\langle \text{previous theorem, using the side condition} \right\rangle \\
& \forall (\tau, \sqsubseteq, s) \in \mathcal{A}_{PBT}\ P\ \rho \quad S(\tau, \sqsubseteq, s) \\
\Leftrightarrow & \left\langle \text{definition of } \mathbf{sat} \text{ in } \mathcal{M}_{PTB} \right\rangle \\
& P\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{PTB}
\end{aligned}$$

as required. □

To prove that a probabilistic process satisfies a hard specification, it is enough to prove that the corresponding uuprobabilistic process satisfies the same specification.

5.2.2 Abstraction from the Deterministic Model

The Deterministic Model sits strictly inside the Prioritized Model so our abstraction mappings $\varphi_D^{(B)} : DTCSP \to BTCSP$ and $\theta_D^{(B)} : \mathcal{M}_{DTB} \to \mathcal{M}_{TB}$ are simply the identity functions.

$$\varphi_D^{(B)}(P) \mathrel{\widehat{=}} P \qquad \theta_D^{(B)}(A) \mathrel{\widehat{=}} A$$

The following theorem was proved in section 3.6:

Theorem 5.2.4: $\theta_D^{(B)}(\mathcal{M}_{DTB}) \subseteq \mathcal{M}_{TB}$. ♡

The following theorem is trivial to prove by structural induction:

Theorem 5.2.5: For all DTCSP processes P and environments ρ,

$$\theta_D^{(B)}(\mathcal{A}_{DT}\ P\ \rho) = \mathcal{A}_{BT}\ \varphi_D^{(B)}(P)\ \rho$$

♡

The following proof rule can be derived from this result in the same way that the proof rule in the previous section was derived from the abstraction result there:

Rule 5.2.6:

$$\frac{\varphi_D^{(B)}(P)\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{TB}}{P\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{DTB}}$$

△

Thus we have shown that proving that a specification holds of a process in either the Probabilistic or Deterministic Model can be reduced to showing that a corresponding process in the (unprobabilistic) Prioritized Model satisfies the same specification. The rest of this chapter will be devoted to methods of showing that a prioritized process satisfies a specification.

5.3 A language for specifying prioritized processes

In order to write readable specifications for prioritized processes we need a specification language; this will be based upon the language described in section 2.5.

5.3.1 Primitive specifications

We write $(a \text{ at } t)(\tau, \sqsubseteq, s)$ to specify that event a occurs at time t:

$$(a \text{ at } t)(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} a \in s(t)$$

As in section 2.5, we generalise this to specify that some event a from a set A occurs at some time t during the interval I:

$$A \text{ at } I \mathrel{\widehat{=}} \exists\, a \in A \quad \exists\, t \in I \quad a \text{ at } t$$

We also generalise the at macro in order to specify that n events from some set occur during some interval:

$$A \ \mathsf{at}^n \ I \mathrel{\widehat=} \#(s \quad A \uparrow I) \quad n$$

And we can specify that events do *not* occur:

$$\begin{aligned} \mathsf{no} \ a \ \mathsf{at} \ t &\mathrel{\widehat=} \neg (a \ \mathsf{at} \ t) \\ \mathsf{no} \ A \ \mathsf{at} \ I &\mathrel{\widehat=} \neg (A \ \mathsf{at} \ I) \\ \mathsf{no} \ A \ \mathsf{at}^n \ I &\mathrel{\widehat=} \neg (A \ \mathsf{at}^n \ I) \end{aligned}$$

We will sometimes want to be able to specify that a process acts in a particular way *if* we have observed it for long enough.

$$(\mathsf{beyond} \ t)(\tau, \sqsubseteq, s) \mathrel{\widehat=} \tau > t$$

We will use the offered macro to specify that a process is willing to perform a particular event:

$$(a \ \mathsf{offered} \ t)(\tau, \sqsubseteq, s) \mathrel{\widehat=} \mathsf{beyond} \ t \Rightarrow (t, a) \in \text{items} \sqsubseteq$$

We can also specify that an event a would be refused at time t if it were offered by the environment in addition to what was performed. This is true if the process does not prefer an extra a in addition to what it performs at t ($s \uparrow t \uplus (t, \{\!|a|\!\})$) to what it performs ($s \uparrow t$). This gives the following definition:

$$(a \ \mathsf{ref} \ t)(\tau, \sqsubseteq, s) \mathrel{\widehat=} \mathsf{beyond} \ t \wedge s \uparrow t \uplus (t, \{\!|a|\!\}) \not\sqsupset s \uparrow t$$

As in the Timed Failures Model, we will not use this predicate directly in specifications: we will use it to define more useful macros.

We can also specify that an event is *not* refused:

$$\mathsf{no} \ a \ \mathsf{ref} \ t \mathrel{\widehat=} \neg (a \ \mathsf{ref} \ t)$$

so

$$(\mathsf{no} \ a \ \mathsf{ref} \ t)(\tau, \sqsubseteq, s) = t \quad \tau \vee s \uparrow t \uplus (t, \{\!|a|\!\}) \sqsupset s \uparrow t$$

And we can generalise both these predicates to sets of events:

$$A \ \mathsf{ref} \ t \mathrel{\widehat=} \forall a \in A \quad a \ \mathsf{ref} \ t \qquad \mathsf{no} \ A \ \mathsf{ref} \ t \mathrel{\widehat=} \forall a \in A \quad \mathsf{no} \ a \ \mathsf{ref} \ t$$

5.3.2 Liveness specifications

Recall the definition of the offered macro:

$$(a \ \mathsf{offered} \ t)(\tau, \sqsubseteq, s) \mathrel{\widehat=} \mathsf{beyond} \ t \Rightarrow (t, a) \in \text{items} \sqsubseteq$$

This generalises to say that the process offers one of a set of events A at time t:

$$A \ \mathsf{offered} \ t \mathrel{\widehat=} \exists a \in A \quad a \ \mathsf{offered} \ t$$

We can also say that an event is offered throughout some interval, until it is performed:

$$a \ \mathsf{offered} \ I \mathrel{\widehat=} \forall t \in I \quad a \ \mathsf{at} \ I \cap [0, t) \vee a \ \mathsf{offered} \ t$$

a offered I is true if at all times t during I, if an a has not yet been observed, then a offered t. It will be useful to say that an event is offered from some time until it is performed:

$$a \text{ offered from } t \mathrel{\widehat{=}} a \text{ offered } [t, \infty)$$

Thus from t is an abbreviation for $[t, \infty)$.

We can also specify that events are *not* offered:

$$\begin{aligned} \text{no } a \text{ offered } t &\mathrel{\widehat{=}} \neg\, a \text{ offered } t \\ \text{no } A \text{ offered } I &\mathrel{\widehat{=}} \forall a \in A \quad \forall t \in I \quad \text{no } a \text{ offered } t \end{aligned}$$

If the set I is omitted, we will take it to be the set of all times:

$$\text{no } A \text{ offered} \mathrel{\widehat{=}} \text{no } A \text{ offered } [0, \infty)$$

The live macro is used to specify that the process is willing to perform an event at a particular time. Its definition is the same as in the Timed Failures Model:

$$a \text{ live } t \mathrel{\widehat{=}} a \text{ at } t \vee \text{no } a \text{ ref } t$$

a live t is true if either an a is performed at time t or it is not refused.

This can be generalised to take a set of events as argnment:

$$A \text{ live } t \mathrel{\widehat{=}} A \text{ at } t \vee \text{no } A \text{ ref } t$$

We can also generalise the live macro to specify that an event is available throughout some interval, until it is performed:

$$a \text{ live } I \mathrel{\widehat{=}} \forall t \in I \quad a \text{ at } I \cap [0, t] \vee \text{no } a \text{ ref } t$$

a live I is true if at all times in I, if an a has not yet been observed, then it is available. This generalises to a set of events in the obvious way:

$$A \text{ live } I \mathrel{\widehat{=}} \forall t \in I \quad A \text{ at } I \cap [0, t] \vee \text{no } A \text{ ref } t$$

It will be particularly useful to be able to specify that an event becomes available at some time t and remains available until performed:

$$a \text{ live from } t \mathrel{\widehat{=}} a \text{ live } [t, \infty) \qquad A \text{ live from } t \mathrel{\widehat{=}} A \text{ live } [t, \infty)$$

We can also specify that a process is able to perform n copies of an event:

$$\begin{aligned} a \text{ live}^n\ t &\mathrel{\widehat{=}} a \text{ at}^n\ t \vee \text{no } a \text{ ref } t \\ A \text{ live}^n\ t &\mathrel{\widehat{=}} A \text{ at}^n\ t \vee \text{no } A \text{ ref } t \\ a \text{ live}^n\ I &\mathrel{\widehat{=}} \forall t \in I \quad a \text{ at}^n\ I \cap [0, t] \vee \text{no } a \text{ ref } t \\ A \text{ live}^n\ I &\mathrel{\widehat{=}} \forall t \in I \quad A \text{ at}^n\ I \cap [0, t] \vee \text{no } A \text{ ref } t \end{aligned}$$

The two macros offered and live are quite closely related. By condition A5 on behaviours we have

$$a \text{ live } t \Rightarrow a \text{ offered } t$$

and by condition A3 we have

$$a \text{ offered } (t_0, t_1] \Rightarrow a \text{ live } [t_0, t_1)$$

so if we restrict ourselves to half-open intervals, the two macros are equivalent:

$$a \text{ live } [t_0, t_1) \Leftrightarrow a \text{ offered } [t_0, t_1)$$

Another specification technique that will prove useful is to say that two events a and b cannot both be offered at the same time:

$$a, b \text{ separate } t \mathrel{\hat=} a \text{ offered } t \Rightarrow \text{no } b \text{ offered } t$$

This can be generalised in the obvious ways:

$$A, B \text{ separate } I \mathrel{\hat=} \forall a \in A \,;\, b \in B \quad \forall t \in I \quad a, b \text{ separate } t$$
$$A_1, \ldots, A_n \text{ separate } I \mathrel{\hat=} \forall i, j : 1 \,..\, n \quad i \neq j \Rightarrow A_i, A_j \text{ separate } I$$

5.3.3 Priorities

We extend our specification language to allow us to specify that certain priorities hold. We write α preferred to β @ t to specify that the process prefers α to β at time t (if we have observed the process until time t):

$$(\alpha \text{ preferred to } \beta \text{ @ } t)(\tau, \sqsubseteq, s) \mathrel{\hat=} \text{beyond } t \Rightarrow (t, \alpha) \sqsupset (t, \beta)$$

If the bags α and β are singletons then we will omit bag brackets to improve readability. We can generalise this to include several preferences:

$$\alpha_0 \text{ preferred to } \alpha_1 \text{ preferred to} \ldots \text{preferred to } \alpha_n \text{ @ } t \mathrel{\hat=}$$
$$\forall i : 0 \,..\, n-1 \quad \alpha_i \text{ preferred to } \alpha_{i+1} \text{ @ } t$$

We generalise this further to specify that certain priorities hold throughout some interval, until one of the events occurs:

$$\alpha \text{ preferred to } \beta \text{ @ } I \mathrel{\hat=} \forall t \in I \quad \alpha \cup \beta \text{ at } I \cap [0, t) \vee \alpha \text{ preferred to } \beta \text{ @ } t$$

$$\alpha_0 \text{ preferred to } \alpha_1 \text{ preferred to} \ldots \text{preferred to } \alpha_n \text{ @ } I \mathrel{\hat=}$$
$$\forall t \in I \quad (\textstyle\bigcup \alpha_i) \text{ at } I \cap [0, t) \vee \alpha_0 \text{ preferred to } \alpha_1 \text{ preferred to} \ldots \text{preferred to } \alpha_n \text{ @ } t$$

$$\alpha_0 \text{ preferred to } \alpha_1 \text{ preferred to} \ldots \text{preferred to } \alpha_n \text{ from } t \mathrel{\hat=}$$
$$\alpha_0 \text{ preferred to } \alpha_1 \text{ preferred to} \ldots \text{preferred to } \alpha_n \text{ @ } [t, \infty)$$

where α at I for bag α has the obvious meaning:

$$\alpha \text{ at } I \mathrel{\hat=} \exists a \in \alpha \quad t \in I \quad a \text{ at } t$$

5.3.4 History predicates

As in section 2.5, we will often want to write specifications of the form $\varphi(M(s))$, where M is a projection function that extracts some information from a trace, and φ is a predicate. In this section we define a few useful projection functions.

The functions first and last return the first or last timed events observed during a behaviour.

$$\mathsf{first}(s) \mathrel{\widehat{=}} \mathit{head}\ s \qquad \mathsf{last}(s) \mathrel{\widehat{=}} \mathit{foot}\ s$$

Note that this is a pair consisting of a time and an *action*: more than one event could have occurred at the same time. These can be qualified with one of the terms before t, after t or during I to restrict attention to a particular set of times. We can also restrict our attention to a particular set of events. For example:

$$\begin{aligned} (\mathsf{first}\ A\ \mathsf{after}\ t)(s) &\mathrel{\widehat{=}} \mathit{head}(s \quad A \quad t) \\ (\mathsf{last}\ A\ \mathsf{before}\ t)(s) &\mathrel{\widehat{=}} \mathit{foot}(s \quad A \quad t) \\ (\mathsf{last\ during}\ I)(s) &\mathrel{\widehat{=}} \mathit{foot}(s \uparrow I) \end{aligned}$$

These operators will allow us to write specifications such as last A before $3 = (2.\{\!|a, b|\!\})$. Omitting bag brackets for singleton actions will make our specifications more readable, for example last A before $3 = (2, a)$.

The functions time of and name of return the time and action components of a timed action:

$$\mathsf{time\ of}\ (t, \alpha) \mathrel{\widehat{=}} t \qquad \mathsf{name\ of}\ (t, \alpha) \mathrel{\widehat{=}} \alpha$$

These functions can be used to write predicates of the form time of first A after $2 \quad 3$ or name of last $A = a$.

Other functions that we will find useful are alphabet which returns the set of (untimed) events observed, and count A which returns the number of events from the set A observed:

$$\mathsf{alphabet}(s) \mathrel{\widehat{=}} \Sigma s \qquad \mathsf{count}\ A(s) \mathrel{\widehat{=}} \#(s \quad A)$$

These can be qualified with the phrases before t, after t or during I; we will omit the argument A of count if we want to refer to the total number of events performed, i.e. in the case $A = \Sigma$.

It will sometimes be useful to say that no events are performed:

$$\mathsf{silent}(s) \mathrel{\widehat{=}} s = \prec\succ$$

This can be qualified in the normal ways, for example

$$(\mathsf{silent\ before}\ t)(s) \mathrel{\widehat{=}} s \quad t = \prec\succ$$

5.3.5 Environmental assumptions

Often we will want to say that a process acts in a particular way *if* the environment satisfies some condition. In this subsection we describe a few macros for placing conditions on the environment. The definitions of these macros are very similar to in the Timed Failures Model.

We will write a open t to specify that the environment is willing to perform an a at time t; it is true if either an a is actually performed, or if the behaviour is consistent with the environment being willing to perform an additional a, but which the process can refuse.

$$a \text{ open } t \mathrel{\widehat{=}} a \text{ at } t \vee a \text{ ref } t$$

a open t is true if an a is either performed or refused at time t. Any such behaviour is consistent with the environment being willing to perform an extra a.

This macro can be extended to sets of events in the obvious way:

$$A \text{ open } t \mathrel{\widehat{=}} A \text{ at } t \vee A \text{ ref } t$$

We will say a open I if the environment is willing to perform an a at any time during I until one is performed:

$$\begin{aligned} a \text{ open } I &\mathrel{\widehat{=}} \forall t \in I \quad a \text{ at } I \cap [0,t] \vee a \text{ ref } t \\ A \text{ open } I &\mathrel{\widehat{=}} \forall t \in I \quad A \text{ at } I \cap [0,t] \vee A \text{ ref } t \end{aligned}$$

As with live, it is useful to have a special form for the interval $[t,\infty)$:

$$\begin{aligned} a \text{ open from } t &\mathrel{\widehat{=}} a \text{ open } [t,\infty) \\ A \text{ open from } t &\mathrel{\widehat{=}} A \text{ open } [t,\infty) \end{aligned}$$

It will also be useful to be able to specify that the environment is able to perform n copies of an event:

$$\begin{aligned} a \text{ open}^n\ t &\mathrel{\widehat{=}} a \text{ at}^n\ t \vee a \text{ ref } t \\ A \text{ open}^n\ t &\mathrel{\widehat{=}} A \text{ at}^n\ t \vee A \text{ ref } t \\ a \text{ open}^n\ I &\mathrel{\widehat{=}} \forall t \in I \quad a \text{ at}^n\ I \cap [0,t] \vee a \text{ ref } t \\ A \text{ open}^n\ I &\mathrel{\widehat{=}} \forall t \in I \quad A \text{ at}^n\ I \cap [0,t] \vee A \text{ ref } t \end{aligned}$$

To specify that the environment is *not* willing to perform an event, we use the closed macro:

$$a \text{ closed } t \mathrel{\widehat{=}} \neg\,(a \text{ at } t)$$

Any behaviour that satisfies this specification will be consistent with the assumption that the environment is not willing to perform an a. Note that this is the same as no a at t: we will restrict the use of closed to environmental assumptions. This macro generalises in the obvious way:

$$A \text{ closed } I \mathrel{\widehat{=}} \forall a \in A \quad \forall t \in I \quad a \text{ closed } t$$

The specification internal A says that the environment is *always* willing to perform as many events from a set A as the process wants. This will occur when the events from A are hidden:

$$(\text{internal } A)(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} s = \Uparrow_{\sqsubseteq}^{\setminus A}(s \setminus A)$$

internal A is true if the process performs as many (or as few) events from A as it wants. In particular it is true if the environment is willing to perform arbitrary many events from A, such as happens when the events of A are hidden. Put another way, there is no offer v such that $v \setminus A = s \uparrow t \setminus A$ and the process would rather have performed v to what it did perform:

$$\forall t \quad v \quad v \setminus A = s \uparrow t \setminus A \wedge v \sqsupset s \uparrow t$$

In particular

$$\begin{aligned}(\mathsf{internal}\, A)(\tau, \sqsubseteq, s) &\Rightarrow \quad a \in A\,;\, t \quad \tau \quad s \uparrow t \uplus (t, \{\!|a|\!\}) \sqsupset s \uparrow t \\ (\mathsf{internal}\, a)(\tau, \sqsubseteq, s) &\Rightarrow \forall t \quad s \uparrow t \uplus (t, \{\!|a|\!\}) \not\sqsupset s \uparrow t\end{aligned}$$

so the process would never have preferred to perform another member of A. If A is a singleton set we will omit set brackets to improve readability.

Note that we do not have the law $\mathsf{internal}\, A = A\ \mathsf{open}^{\infty}\ [0, \tau]$. Consider the process $b \longleftarrow (a \,\Box\, c) \setminus c$. This initially has an offer relation with $\{\!|b|\!\} \sqsupset \{\!|a, b|\!\} \sqsupset \{\!||\!\} \sqsupset \{\!|a|\!\}$. Suppose it performs $\{\!|a, b|\!\}$ at time 0. Then this behaviour satisfies $a\ \mathsf{open}^{\infty}\ [0, \tau]$, since it will refuse an extra a, but it does not satisfy $\mathsf{internal}\, a$ since it would rather perform one fewer a. However, we do have $\mathsf{internal}\, A \Rightarrow A\ \mathsf{open}^{\infty}\ [0, \tau]$.

It is worth noting that $\mathsf{internal}\, A \wedge \mathsf{internal}\, B \not\Rightarrow \mathsf{internal}(A \cup B)$. Consider the process $(c \,\Box\, (a \longleftarrow b)) \setminus c$. Initially this has an offer relation with $\{\!||\!\} \sqsupset \{\!|a, b|\!\} \sqsupset \{\!|a|\!\} \sqsupset \{\!|b|\!\}$. Suppose it performs $\{\!|a, b|\!\}$ at time 0. Then it satisfies $\mathsf{internal}\{a\} \wedge \mathsf{internal}\{b\}$, but it doesn't satisfy $\mathsf{internal}\{a, b\}$ since $(0, \{\!||\!\}) \setminus \{a, b\} = s \uparrow 0 \setminus \{a, b\}$ and $(0, \{\!||\!\}) \sqsupset s \uparrow 0$.

Note though that we do have the law $\mathsf{internal}(A \cup B) \Rightarrow \mathsf{internal}\, A \wedge \mathsf{internal}\, B$.

Another useful specification technique is to say that the environment is willing to perform a particular bag α. An observation is compatible with this if α is not offered stronger that what is performed:

$$(\alpha\ \mathsf{accessible}\ t)(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} (t, \alpha) \not\sqsupset s \uparrow t$$

$(\alpha\ \mathsf{accessible}\ t)(\tau, \sqsubseteq, s)$ is true if the offer relation of the process does not have (t, α) stronger than $s \uparrow t$. This fits with our intuitions because if the environment is willing to perform α, then we should not have $(t, \alpha) \sqsupset s \uparrow t$, or else the process would have performed α in preference to $s \uparrow t$.

This specification macro can be generalised to say that an action α is offered by the environment throughout some interval unless an event from α is performed:

$$\alpha\ \mathsf{accessible}\ I \mathrel{\widehat{=}} \forall t \in I \quad \alpha\ \mathsf{at}\ I \cap [0, t) \vee (\alpha\ \mathsf{accessible}\ t)$$

As normal, it is useful to specify that an α is available from some time t until it is performed:

$$\alpha\ \mathsf{accessible\ from}\ t \mathrel{\widehat{=}} \alpha\ \mathsf{accessible}\ [t, \infty)$$

We can also generalise to specify that a set of actions is offered by the environment:

$$A\ \mathsf{accessible}\ I \mathrel{\widehat{=}} \forall \alpha \in A \quad \alpha\ \mathsf{accessible}\ I$$

The specification macros '$\mathsf{internal}\, a$' and '$a\ \mathsf{accessible}\ [0, \infty)$' are subtly different: consider a behaviour with $(0, \{\!|a|\!\}) \sqsupset (0, \{\!|b|\!\})$ where a b is performed at time 0. This satisfies the specification $\mathsf{internal}\, a$ but not $a\ \mathsf{accessible}\ 0$. The specification $\mathsf{internal}\, a$ states that no more as could be performed by the process; the specification $a\ \mathsf{accessible}\ 0$ states that $\{\!|a|\!\}$ is not offered stronger than what is performed. The following lemma relates these two concepts and will be useful in later sections.

Lemma 5.3.1: Let $\sqsubseteq\ = \ \sqsubseteq_P {}^{A}\!\!\parallel\!\!^{B} \sqsubseteq_Q$ and $c \in C \subseteq A \cap B$. Then if $(c \text{ live } t)(\tau, \sqsubseteq_Q, s \quad B)$ and $(\text{internal } C \wedge \{c\}, A \setminus C \text{ separate } t)(\tau, \sqsubseteq, s)$ then $(c \text{ accessible } t)(\tau, \sqsubseteq_P, s \quad A)$. ♡

If the events of C are internal, and the slave of a parallel composition is willing to perform $c \in C$ at t then, under certain circumstances, the master is in an environment that is willing to perform a c at t. The circumstances are that if the process can perform a c then it cannot perform any event from $A \setminus C$.

Proof: Assume the premises. From $(c \text{ live } t)(\tau, \sqsubseteq_Q, s \quad B)$ we have

$$c \in \Sigma(s \uparrow t \quad B) \ \vee \ s \uparrow t \quad B \uplus (t, \{\!|c|\!\}) \sqsupset_Q s \uparrow t \quad B$$

so in either case we have

$$s \uparrow t \quad B \setminus C \uplus (t, \{\!|c|\!\}) \in \text{items} \sqsubseteq_Q \qquad (*)$$

using condition A5.

Suppose for a contradiction that $\neg\, (c \text{ accessible } t)(\tau, \sqsubseteq_P, s \quad A)$. Then

$$(t, \{\!|c|\!\}) \sqsupset_P s \uparrow t \quad A$$

and so $\Sigma(s \uparrow t \quad A \setminus C) = \{\}$ because $\{c\}, A \setminus C$ separate t. Define v by

$$v \mathrel{\widehat{=}} s \uparrow t \setminus C \uplus (t, \{\!|c|\!\})$$

Then $v \quad A = (t, \{\!|c|\!\}) \sqsupset_P s \uparrow t \quad A$ and $v \quad B = s \uparrow t \quad B \setminus C \uplus (t, \{\!|c|\!\}) \in \text{items} \sqsubseteq_Q$ by $(*)$; hence $v \sqsupset s \uparrow t$ by the definition of parallel composition of offer relations. Also $v \setminus C = s \uparrow t \setminus C$, contradicting the definition of internal C. □

We have the following corollary.

Corollary 5.3.2: Let $\sqsubseteq\ = \ \sqsubseteq_P {}^{A}\!\!\parallel\!\!^{B} \sqsubseteq_Q$ and $c \in C \subseteq A \cap B$. Then if $(c \text{ live } I)(\tau, \sqsubseteq_Q, s \quad B)$ and $(\text{internal } C \wedge \{c\}, A \setminus C \text{ separate } I)(\tau, \sqsubseteq, s)$ then $(c \text{ accessible } I)(\tau, \sqsubseteq_P, s \quad A)$. ♡

The internal macro tends to be of use when events are hidden. In section 5.4.8 we will show that if we can prove P **sat** internal $A \Rightarrow S(\tau, \sqsubseteq, s)$ then, under certain circumstances, we can deduce that $P \setminus A$ **sat** $S(\tau, \sqsubseteq, s)$. The accessible macro is often introduced when we consider parallel composition, as shown by the above lemma.

The specification language presented in this section is very similar to the specification language presented in section 2.5. In chapter 6 we will show that if

- S is a piece of syntax in the specification language satisfying certain properties, for example if it is made up of ats and lives (without any mention of priorities), combined using conjunctions, implications and negations; *and*
- we can find an unprioritized TCSP process P such that P satisfies the failures specification represented by S,

then any BTCSP refinement of P will satisfy the specification represented by S in the prioritized model. This will allow us to refine processes from the Failures Model into the Prioritized Model.

5.4 Derivation of the proof rules

In this section we derive a complete proof system for behavioural specifications on prioritized processes. The proofs follow very closely those of Schneider [Sch90] and Davies [Dav91] for the proof rules in the Timed Failures Model. The rules are summarized in appendix B.1.

5.4.1 Auxiliary rules

The following rules can be proved directly from the definition of the $\mathbf{sat}_\rho$ relation.

$$\frac{}{P\ \mathbf{sat}_\rho\ true} \qquad \frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S \\ P\ \mathbf{sat}_\rho\ T \end{array}}{P\ \mathbf{sat}_\rho\ S \wedge T} \qquad \frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S \\ S(\tau, \sqsubseteq, s) \Rightarrow T(\tau, \sqsubseteq, s) \end{array}}{P\ \mathbf{sat}_\rho\ T}$$

Every process satisfies the null specification; if a process satisfies two predicates, then it satisfies their conjunction; and if a process satisfies some specification, then it satisfies any weaker specification.

5.4.2 Basic processes

The semantic equations for the basic processes $STOP$, $SKIP$ and $WAIT\ t$ are all of the form

$$\mathcal{A}_{BT}\ P\ \rho \mathrel{\hat{=}} \{b \mid T(b)\}$$

The corresponding proof rule is of the form

$$\frac{T(b) \Rightarrow S(b)}{P\ \mathbf{sat}_\rho\ S(b)}$$

This is sound since, from the semantic equation, $\forall\, b \in \mathcal{A}_{BT}\ P\ \rho \quad T(b)$; then from the premise, $\forall\, b \in \mathcal{A}_{BT}\ P\ \rho \quad S(b)$, so $P\ \mathbf{sat}_\rho\ S(b)$.

5.4.3 Unary operators

The semantic equations for the unary operators prefixing, delay, abstraction, and renaming can all be written in the form

$$\mathcal{A}_{BT}\ F(P)\ \rho \mathrel{\hat{=}} \{b \mid T'(b)\} \cup \{C(b) \mid f(b) \in \mathcal{A}_{BT}\ P\ \rho \wedge T(b)\}$$

The corresponding proof rule is of the form

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S'(b) \\ T'(b) \Rightarrow S(b) \\ S'(f(b)) \wedge T(b) \Rightarrow S(C(b)) \end{array}}{F(P)\ \mathbf{sat}_\rho\ S(b)}$$

In the cases of hiding and renaming, $T'(b)$ is *false* and so the corresponding antecedent can be dropped. The rule can be proved sound as follows. Assume the antecedents of the rule; then

$$
\begin{array}{ll}
 & b \in \mathcal{A}_{BT}\ F(P)\ \rho \\
\Rightarrow & \langle \text{semantic definition} \rangle \\
 & T'(b) \vee \exists\, b' \quad b = C(b') \wedge f(b') \in \mathcal{A}_{BT}\ P\ \rho \wedge T(b') \\
\Rightarrow & \langle \text{first and second premises} \rangle \\
 & S(b) \vee \exists\, b' \quad b = C(b') \wedge S'(f(b')) \wedge T(b') \\
\Rightarrow & \langle \text{third premise} \rangle \\
 & S(b) \vee \exists\, b' \quad b = C(b') \wedge S(C(b')) \\
\Rightarrow & \langle \text{predicate calculus} \rangle \\
 & S(b)
\end{array}
$$

So $\forall\, b \in \mathcal{A}_{BT}\ F(P)\ \rho \quad S(b)$, i.e. $F(P)\ \mathbf{sat}_\rho\ S(b)$.

5.4.4 Binary operators

The semantic definitions for the binary operators may be written in the following form:

$$\mathcal{A}_{BT}\ P \oplus Q\ \rho \mathrel{\widehat=} \{C(b) \mid f_P(b_P) \in \mathcal{A}_{BT}\ P\ \rho \wedge f_Q(b_Q) \in \mathcal{A}_{BT}\ Q\ \rho \wedge R(b, b_P, b_Q)\}$$

The corresponding proof rule is of the form

$$
\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S_P(b) \\ Q\ \mathrm{sat}_\rho\ S_Q(b) \\ S_P(f_P(b_P)) \wedge S_Q(f_Q(b_Q)) \wedge R(b, b_P, b_Q) \Rightarrow S(C(b)) \end{array}}{P \oplus Q\ \mathbf{sat}_\rho\ S(b)}
$$

The rule may be proved as follows. Assume the antecedents hold. Then,

$$
\begin{array}{ll}
 & b' \in \mathcal{A}_{BT}\ P \oplus Q\ \rho \\
\Rightarrow & \langle \text{semantic definition} \rangle \\
 & \exists\, b_P, b_Q, b \quad f_P(b_P) \in \mathcal{A}_{BT}\ P\ \rho \wedge f_Q(b_Q) \in \mathcal{A}_{BT}\ Q\ \rho \wedge R(b, b_P, b_Q) \wedge b' = C(b) \\
\Rightarrow & \langle \text{first and second premises} \rangle \\
 & \exists\, b_P, b_Q, b \quad S_P(f_P(b_P)) \wedge S_Q(f_Q(b_Q)) \wedge R(b, b_P, b_Q) \wedge b' = C(b) \\
\Rightarrow & \langle \text{third premise} \rangle \\
 & \exists\, b_P, b_Q, b \quad S(C(b)) \wedge b' = C(b) \\
\Rightarrow & \langle \text{predieate calculus} \rangle \\
 & S(b')
\end{array}
$$

So $\forall\, b' \in \mathcal{A}_{BT}\ P \oplus Q\ \rho \quad S(b')$, i.e. $P \oplus Q\ \mathbf{sat}_\rho\ S(b)$.

5.4.5 Indexed operators

The semantic equations for the two indexed choice operators, $\ _{i \in I} P_i$ and $c.i : I \xrightarrow{t_i} P_i$ can be written in the form

$$\mathcal{A}_{BT}\ \oplus_{i \in I} P_i\ \rho \mathrel{\widehat=} \{b \mid T'(b)\} \cup \{C(b) \mid \exists\, i \in I \quad f(b) \in \mathcal{A}_{BT}\ P_i\ \rho \wedge T(b)\}$$

The corresponding proof rule is

$$\frac{\begin{array}{l}\forall i \in I \quad P_i \ \mathbf{sat}_\rho \ S_i(b) \\ T'(b) \Rightarrow S(b) \\ \forall i \in I \quad S_i(f(b)) \wedge T(b) \Rightarrow S(C(b))\end{array}}{\oplus_{i\in I} P_i \ \mathbf{sat}_\rho \ S(b)}$$

In the case of infinite nondeterministic choice, the predicate $T'(b)$ is *false*, so the corresponding premise in the inference rule is dropped. The rule can be proved sound as follows. Assume the premises of the proof rule hold. Then we have

$$\begin{array}{ll} & b \in \mathcal{A}_{BT} \ \oplus_{i\in I} P_i \ \rho \\ \Rightarrow & \left\langle \text{semantic definition} \right\rangle \\ & T'(b) \vee \exists\, b' \quad b = C(b') \wedge \exists\, i \in I \quad f(b') \in \mathcal{A}_{BT} \ P_i \ \rho \wedge T(b') \\ \Rightarrow & \left\langle \text{premises 1 and 2} \right\rangle \\ & S(b) \vee \exists\, b' \quad b = C(b') \wedge \exists\, i \in I \quad S_i(f(b')) \wedge T(b') \\ \Rightarrow & \left\langle \text{premise 3; predicate calculus} \right\rangle \\ & S(b) \vee \exists\, b' \quad b = C(b') \wedge S(C(b')) \\ \Rightarrow & \left\langle \text{predicate calculus} \right\rangle \\ & S(b) \end{array}$$

Hence, $\forall\, b \in \mathcal{A}_{BT} \ \oplus_{i\in I} P_i \ \rho \quad S(b)$, so $\oplus_{i\in I} P_i \ \mathbf{sat}_\rho \ S(b)$.

5.4.6 Recursion

In order to derive a proof rule for recursion, we reason about the topological space on which the model is based. The following theorem is taken from [Ros82]:

Theorem 5.4.1: Let $M = (A, d)$ be a complete metric space, and let TV be the topological space $(\{true, false\}, \mathcal{T})$ where $\mathcal{T} \mathrel{\widehat{=}} \{\{\}, \{false\}, \{true, false\}\}$. If:

- $F : M \to T$ is continuous with respect to the d-open topology and $\mathcal{T}$,
- the set $\{a \in A \mid F(a) = true\}$ is non-empty,
- the function $C : M \to M$ is a contraction mapping, *and*
- $\forall\, x : A \quad F(x) = true \Rightarrow F(C(x)) = true$,

then $F(fix(C)) = true$. ♡

We define a predicate to be satisfiable if it is satisfied by some element of $\mathcal{S}_{TB}$:

Definition 5.4.2: The predicate R is satisfiable if $\exists\, A : \mathcal{S}_{TB} \quad R(A)$. ◇

In the following subsections we prove the soundness of the proof rules for immediate recursion, delayed recursion and mutual recursion.

Immediate recursion

If P is constructive for X then we have the following proof rule for immediate recursion:

Rule 5.4.3:

$$\frac{\forall Y \cdot \mathcal{S}_{TB} \quad R(Y) \Rightarrow R(\mathcal{A}_{BT} \quad P \quad \rho[Y/X])}{R(\mathcal{A}_{BT} \quad \mu X \quad P \quad \rho)} \Big[R \text{ continnous and satisfiable} \Big]$$

$\triangle$

Proof: If P is constrnctive for X then tbe mapping $\lambda X \quad \mathcal{A}_{BT} \quad P \quad \rho[Y/X]$ is a contraction mapping. By hypothesis, R is continuous and $\{A : \mathcal{S}_{TB} \mid R(A) = true\}$ is non-empty. We have assumed that

$$\forall Y : \mathcal{S}_{TB} \quad R(Y) \Rightarrow R(\mathcal{A}_{BT} \quad P \quad \rho[Y/X])$$

Hence we may apply theorem 5.4.1 to show that the result holds. $\square$

We are only interested in behavioural specifications; this allows onr proof rule to be simplified:

Rule 5.4.4:

$$\frac{X \ \mathbf{sat}_\rho \ S \Rightarrow P \ \mathbf{sat}_\rho \ S}{\mu X \quad P \ \mathbf{sat}_\rho \ S}$$

$\triangle$

We need the following result adapted from [Ree88]:

Theorem 5.4.5: A specification R is continuous if for all X in $\mathcal{S}_{TB}$ such that $R(X) = false$:

$$\exists t : TIME \quad \forall Y : \mathcal{S}_{TB} \quad Y \quad t = X \quad t \Rightarrow R(Y) = false$$

$\heartsuit$

We can now prove the inference rule sound:

Proof: In order to use rule 5.4.3 we only need to prove that the predicate

$$R(Y) \mathrel{\hat{=}} \forall (\tau, \sqsubseteq, s) \in Y \quad S(\tau, \sqsubseteq, s)$$

is continuous and satisfiable. It is satisfiable since $R(\{\})$ obviously holds. To show continuity, suppose that $X \in \mathcal{S}_{TB}$ and $R(X) = false$. Then $\exists (\tau, \sqsubseteq, s) \in X \quad \neg\, S(\tau, \sqsubseteq, s)$. Pick $t \quad \tau$. Then for all $Y \in \mathcal{S}_{TB}$:

$$Y \quad t = X \quad t \Rightarrow (\tau, \sqsubseteq, s) \in Y$$

But $\neg\, S(\tau, \sqsubseteq, s)$, so $R(Y) = false$, as required. $\square$

Note that in proving $X \ \mathbf{sat}_\rho \ S \Rightarrow P \ \mathbf{sat}_\rho \ S$ we cannot assnme that X is a member of $\mathcal{M}_{TB}$: we may not assume that any of the axioms are satisfied by X. This is rarely a problem.

Delayed recursion

The following proof rule holds for the delayed recursion operator:

Rule 5.4.6:

$$\frac{X \ \mathbf{sat}_\rho \ (S((\tau, \sqsubseteq, s) - \delta) \wedge \mathit{begin}\ s \quad \delta \wedge \sqsubseteq \quad \delta = [0, \delta) \otimes \langle\{\!|\,|\!\}\rangle) \Rightarrow P \ \mathbf{sat}_\rho \ S(\tau, \sqsubseteq, s)}{\mu X \quad P \ \mathbf{sat}_\rho \ S(\tau, \sqsubseteq, s)}$$

$\triangle$

Proof: The proof of this is similar to the proof of the rule for immediate recursion. $\Box$

Mutual recursion

We restrict our attention to recursive equation sets that have a vector of terms that is constructive for the vector of variables. The following rule shows that if a vector of closed, satisfiable predicates $\underline{R}$ is preserved by the semantic mapping, then it is satisfied by the fixed point.

Rule 5.4.7:

$$\frac{(\forall i \quad R_i(Y_i)) \Rightarrow \forall j \quad R_j(\mathcal{A}_{BT} \ P_j \ \rho[\underline{Y}/\underline{X}])}{\forall j \quad R_j(\mathcal{A}_{BT} \ \langle X_i = P_i \rangle_j \ \rho)} \left[R_i \text{ closed, satisfiable} \right]$$

$\triangle$

Proof: Recall that in the proof of soundness for mutual recursion, described in chapter 4, we defined a secondary vector of processes $\underline{Q}$ by

$$Q_i \mathrel{\widehat{=}} P_i[Q_j/X_j \mid j \in \mathit{seg}(i)]$$

We defined $M(\underline{X}, \underline{P})\rho$ by $M(\underline{X}, \underline{P})\rho \mathrel{\widehat{=}} \lambda \underline{Y} \quad \mathcal{A}_{BT} \ \underline{P} \ \rho[\underline{Y}/\underline{X}]$, and defined $M(\underline{X}, \underline{Q})\rho$ similarly. We showed that $M(\underline{X}, \underline{Q})\rho$ is a contraction mapping, and that its unique fixed point is also the unique fixed point of $M(\underline{X}, \underline{P})\rho$.

Assume, then, the premise and side condition of the proof rule. We claim that

$$(\forall i : I \quad R_i(Y_i)) \Rightarrow \forall j \quad R_j(\mathcal{A}_{BT} \ Q_j \ \rho[\underline{Y}/\underline{X}])$$

we prove this by transfinite induction. Define J by

$$J \mathrel{\widehat{=}} \{k : I \mid (\forall i : I \quad R_i(Y_i)) \Rightarrow R_k(\mathcal{A}_{BT} \ Q_k \ \rho[\underline{Y}/\underline{X}])\}$$

We assume $\mathit{seg}(k) \subseteq J$ and prove that $k \in J$. Assume that

$$\forall i : I \quad R_i(Y_i) \qquad (*)$$

Then by definition of $\underline{Q}$,

$$\mathcal{A}_{BT}\ Q_k\ \rho[\underline{Y}/\underline{X}] = \mathcal{A}_{BT}\ P_k\ \rho[\underline{Y}/\underline{X}][\mathcal{A}_{BT}\ Q_l\ \rho[\underline{Y}/\underline{X}]/X_l \mid l \in seg(k)]$$

Define the vector $\underline{Z}$ by

$$Z_l \mathrel{\hat=} \begin{cases} Y_l & \text{if } l \notin seg(k) \\ \mathcal{A}_{BT}\ Q_l\ \rho[\underline{Y}/\underline{X}] & \text{if } l \in seg(k) \end{cases}$$

Then

$$\mathcal{A}_{BT}\ Q_k\ \rho[\underline{Y}/\underline{X}] = \mathcal{A}_{BT}\ P_k\ \rho[\underline{Z}/\underline{X}]$$

Now, by the inductive hypothesis and $(*)$, $\forall l : I \quad R_l(Z_l)$, so from the premise of the proof rule, $R_k(\mathcal{A}_{BT}\ P_k\ \rho[\underline{Z}/\underline{X}])$, i.e. $R_k(\mathcal{A}_{BT}\ Q_k\ \rho[\underline{Y}/\underline{X}])$. This proves our claim, and shows that $M(\underline{X}, \underline{Q})\rho$ preserves $\underline{R}$.

Now, each R_i is closed and satisfiable, so the vector of predicates $\underline{R}$ is closed and satisfiable. Hence we may apply theorem 5.4.1 to deduce that $\underline{R}$ is satisfied by the fixed point of $M(\underline{X}, \underline{Q})\rho$. But this fixed point is the same as the fixed point of $M(\underline{X}, \underline{P})\rho$, so we deduce that the rule is sound. □

We can use this rule to derive the following rule for behavioural specifications:

Rule 5.4.8:

$$\frac{(\forall i : I \quad X_i\ \mathbf{sat}_\rho\ S_i) \Rightarrow \forall j : I \quad P_i\ \mathbf{sat}_\rho\ S_i}{\langle X_i = P_i \rangle_j\ \mathbf{sat}_\rho\ S_j}$$

△

This rule follows from the previous rule in the same way rule 5.4.4 followed from rule 5.4.3.

5.4.7 Completeness

We claim that the proof system is complete in the sense that if some behavioural specification $S(\tau, \sqsubseteq, s)$ is true of all behaviours of a process P, then the inference rules given in this chapter are sufficient to prove that P **sat** S. We proceed via the following lemma:

Lemma 5.4.9: If $P \in BTCSP$ is such that every recursion is constructive, then we may use the proof rules to establish

$$P\ \mathbf{sat}_\rho\ (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ P\ \rho$$

for any environment ρ. ♡

Proof: We proceed by a structural induction upon the syntax of BTCSP. The result is easily established for the base cases. For example, consider the process $STOP$. The inference rule

$$\frac{S(\tau, [0,\tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec\succ)}{STOP\ \mathbf{sat}_\rho\ S}$$

allows us to establish that $STOP\ \mathbf{sat}_\rho \sqsubseteq = [0,\tau] \otimes \langle\{\!\}\!\}\rangle \wedge s = \prec\succ$. From the semantic definition we have that

$$\sqsubseteq = [0,\tau] \otimes \langle\{\!\}\!\}\rangle \wedge s = \prec\succ \Rightarrow (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ STOP\ \rho$$

So the inference rule

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S \\ S(\tau,\sqsubseteq,s) \Rightarrow T(\tau,\sqsubseteq,s) \end{array}}{P\ \mathbf{sat}_\rho\ T}$$

allows us to establish that $STOP\ \mathbf{sat}_\rho\ (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ STOP\ \rho$

For composite processes we assume the result holds for the subcomponents, and apply the appropriate proof rule. For example, consider the left-biased lockstep parallel operator. By induction, we know that the proof rules are strong enough to prove

$$\begin{array}{l} P\ \ \mathbf{sat}_\rho\ \ (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ P\ \rho \\ Q\ \ \mathbf{sat}_\rho\ \ (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ Q\ \rho \end{array}$$

The semantic equation for $P \Lsh\!\!\!\Vert Q$ gives us that

$$(\tau,\sqsubseteq_P,s) \in \mathcal{A}_{BT}\ P\ \rho \wedge (\tau,\sqsubseteq_Q,s) \in \mathcal{A}_{BT}\ Q\ \rho \Rightarrow (\tau,\sqsubseteq_P \Lsh\!\!\!\Vert \sqsubseteq_Q,s) \in \mathcal{A}_{BT}\ P \Lsh\!\!\!\Vert Q\ \rho$$

Then the inference rule

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S_P \\ Q\ \mathbf{sat}_\rho\ S_Q \\ S_P(\tau,\sqsubseteq_P,s) \wedge S_Q(\tau,\sqsubseteq_Q,s) \Rightarrow S(\tau,\sqsubseteq_P \Lsh\!\!\!\Vert \sqsubseteq_Q,s) \end{array}}{P \Lsh\!\!\!\Vert Q\ \mathbf{sat}_\rho\ S}$$

Instantiated with

$$\begin{array}{rcl} S_P(\tau,\sqsubseteq,s) & \mathrel{\hat=} & (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ P\ \rho \\ S_Q(\tau,\sqsubseteq,s) & \mathrel{\hat=} & (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ Q\ \rho \\ S(\tau,\sqsubseteq,s) & \mathrel{\hat=} & (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ P \Lsh\!\!\!\Vert Q\ \rho \end{array}$$

allows us to prove

$$P \Lsh\!\!\!\Vert Q\ \mathbf{sat}_\rho\ (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ P \Lsh\!\!\!\Vert Q\ \rho$$

as required.

For recursion we prove the result for the immediate recursion operator; the other types of recursion are similar. Recall that the semantics of $\mu X \quad P$ is defined to be the unique fixed point of the mapping $M(X,P)\rho$ where

$$M(X,P)\rho \mathrel{\hat=} \lambda Y \quad \mathcal{A}_{BT}\ P\ \rho[Y/X]$$

Let $S(\tau,\sqsubseteq,s) \mathrel{\hat=} (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ \mu X \quad P\ \rho$; we will show

$$X\ \mathbf{sat}_\rho\ S \Rightarrow P\ \mathbf{sat}_\rho\ S$$

Assume X $\mathbf{sat}_\rho$ S. Then $\forall(\tau, \sqsubseteq, s) \in \rho\ X \quad (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ \mu X \quad P\ \rho$, i.e.

$$\rho\ X\ \subseteq \mathcal{A}_{BT}\ \mu X \quad P\ \rho$$

Because of the way the semantics for each operator is defined, the mapping on $\mathcal{M}_{TB}$ corresponding to any BTCSP term is monotonic with respect to the subset relation. So

$$M(X,P)\rho(\rho\ X\ ) \subseteq M(X,P)\rho(\mathcal{A}_{BT}\ \mu X \quad P\ \rho)$$

Hence, expanding the definition of $M(X,P)\rho$, we have

$$\begin{aligned}
& \mathcal{A}_{BT}\ P\ \rho[\rho\ X\ /X] \subseteq M(X,P)\rho(\mathcal{A}_{BT}\ \mu X \quad P\ \rho) \\
\Rightarrow\ & \left\langle \text{definition of substitution} \right\rangle \\
& \mathcal{A}_{BT}\ P\ \rho \subseteq M(X,P)\rho(\mathcal{A}_{BT}\ \mu X \quad P\ \rho) \\
\Rightarrow\ & \left\langle \mathcal{A}_{BT}\ \mu X \quad P\ \rho \text{ is the fixed point of } M(X,P)\rho \right\rangle \\
& \mathcal{A}_{BT}\ P\ \rho \subseteq \mathcal{A}_{BT}\ \mu X \quad P\ \rho \\
\Rightarrow\ & \left\langle \text{definition of } \mathbf{sat}_\rho \right\rangle \\
& P\ \mathbf{sat}_\rho\ (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ \mu X \quad P\ \rho
\end{aligned}$$

So P $\mathbf{sat}_\rho$ S. Hence we can use the proof rule

$$\frac{X\ \mathbf{sat}_\rho\ S \Rightarrow P\ \mathbf{sat}_\rho\ S}{\mu X \quad P\ \mathbf{sat}_\rho\ S}$$

to infer that $\mu X \quad P\ \mathbf{sat}_\rho\ (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ \mu X \quad P\ \rho$ as required. This concludes the proof $\square$

We have shown that our proof rules are enough to establish $P\ \mathbf{sat}_\rho\ (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ P\ \rho$. If a specification $S(\tau, \sqsubseteq, s)$ holds of a process P, then $(\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ P\ \rho \Rightarrow S(\tau, \sqsubseteq, s)$. Then the proof rule

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S' \\ S'(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{P\ \mathbf{sat}_\rho\ S}$$

with $S'(\tau, \sqsubseteq, s)$ instantiated with $(\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ P\ \rho$ can be used to prove that P $\mathbf{sat}_\rho$ S.

5.4.8 Hiding

In this subsection we consider a way of simplifying the rule for hiding. We define a specification S to be A-independent if the removal of A's events from the trace and offer relation does not affect the truth of S.

Definition 5.4.10: A behavioural specification S is A-independent iff

$$\forall(\tau, \sqsubseteq, s) \quad S(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq \setminus A, s \setminus A)$$

$\diamond$

We have the following inference rule:

Rule 5.4.11:

$$\frac{P\ \mathbf{sat}_\rho\ \mathsf{internal}\,A \Rightarrow S}{P \setminus A\ \mathbf{sat}_\rho\ S}\left[S \text{ is } A\text{-independent}\right]$$

$\triangle$

If S is A-independent and $P\ \mathbf{sat}_\rho\ \mathsf{internal}\,A \Rightarrow S$ then we can deduce $P \setminus A\ \mathbf{sat}_\rho\ S$.

Proof: Assume the premise and the side condition. Then we have

$$\begin{aligned}
& (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ P \setminus A\ \rho \\
\Rightarrow\ & \left\langle \text{semantic definition} \right\rangle \\
& \exists \sqsubseteq', s' \quad \sqsubseteq' \setminus A = \sqsubseteq \wedge s' \setminus A = s \wedge s' = \Uparrow_{\sqsubseteq'}^{\sim\setminus A}(s' \setminus A) \wedge (t, \sqsubseteq', s') \in \mathcal{A}_{BT}\ P\ \rho \\
\Rightarrow\ & \left\langle \text{from the premise, definition of } \mathsf{internal}\,A \right\rangle \\
& \exists \sqsubseteq', s' \quad \sqsubseteq' \setminus A = \sqsubseteq \wedge s' \setminus A = s \wedge S(\tau, \sqsubseteq', s') \\
\Rightarrow\ & \left\langle S \text{ is } A\text{-independent} \right\rangle \\
& S(\tau, \sqsubseteq, s)
\end{aligned}$$

Hence $\forall (\tau, \sqsubseteq, s) \in \mathcal{A}_{BT}\ P \setminus A\ \rho \quad S(\tau, \sqsubseteq, s)$, i.e. $P \setminus A\ \mathbf{sat}_\rho\ S$. $\square$

5.4.9 Arguing about probabilistic processes

Up until now we have been discussing proof rules for unprobabilistic prioritized processes. If we want to prove that an unprobabilistic specification is met by a probabilistic process, then we can use the abstraction result presented in section 5.2 to reduce the proof obligation to proving a specification on a BTCSP process, and then apply the proof rules for the unprobabilistic, prioritized model.

Alternatively, we can derive proof rules for arguing directly about probabilistic processes. The proof rules for unprobabilistic operators take precisely the same form as in the unprobabilistic model. For example, we have the following rule for lockstep parallel composition:

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S_P \text{ in } \mathcal{M}_{PTB} \\ Q\ \mathbf{sat}_\rho\ S_Q \text{ in } \mathcal{M}_{PTB} \\ S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \,\#\, \sqsubseteq_Q, s) \end{array}}{P \,\#\, Q\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{PTB}}$$

This can be proved as follows. Recall that in section 5.2 we proved the following result:

$$\varphi_P^{(B)}(P)\ \mathbf{sat}_{\rho'}\ S \text{ in } \mathcal{M}_{TB} \Leftrightarrow P\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{PTB} \qquad \text{if } \rho' = \pi_1 \circ \rho \tag{$*$}$$

Assume the premises of the above rule. Then if $\rho' = \pi_1 \circ \rho$ we have

$$\varphi_P^{(B)}(P)\ \mathbf{sat}_{\rho'}\ S_P \text{ in } \mathcal{M}_{TB} \qquad \varphi_P^{(B)}(Q)\ \mathbf{sat}_{\rho'}\ S_Q \text{ in } \mathcal{M}_{TB}$$

$\vdots$

by $(*)$. Applying the proof rule for parallel composition in $\mathcal{M}_{TB}$, we have

$$\varphi_P^{(B)}(P) \parallel \varphi_P^{(B)}(Q) \;=\; \varphi_P^{(B)}(P \parallel Q)\ \mathbf{sat}_{\rho'}\ S \text{ in } \mathcal{M}_{TB}$$

so $P \parallel Q\ \mathbf{sat}_\rho\ S$ in $\mathcal{M}_{PTB}$, by $(*)$ again.

The probabilistic internal choice operators have the following rules:

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S_P \\ Q\ \mathbf{sat}_\rho\ S_Q \\ S_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{P\ {}_p\sqcap_q\ Q\ \mathbf{sat}_\rho\ S} \qquad \frac{\begin{array}{l} \forall i \in I \quad P_i\ \mathbf{sat}_\rho\ S_i \\ \forall i \in I \quad S_i(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{{}_{i \in I}[p_i]P_i\ \mathbf{sat}_\rho\ S}$$

These can be proved in exactly the same way as the rule for parallel composition, above.

For probabilistic external choice we have the following rule:

$$\frac{\begin{array}{l} P \mathbin{◧} Q\ \mathbf{sat}_\rho\ S \\ P \mathbin{◨} Q\ \mathbf{sat}_\rho\ S \end{array}}{P\ {}_p\ {}_q\ Q\ \mathbf{sat}_\rho\ S}$$

This can be proved using the proof rule for binary probabilistic internal choice and the fact that $P\ {}_p\ {}_q\ Q$ is by definition equal to $P \mathbin{◧} Q\ {}_p\sqcap_q\ P \mathbin{◨} Q$.

Thus, we can prove a probabilistic specification holds of a process either by applying the abstraction result and arguing in $\mathcal{M}_{TB}$, or by applying these inference rules for probabilistic processes directly.

5.5 Example: the lift system revisited

In this section we consider the lift system that was introduced in section 3.2. Recall that the lift system was defined by

$$\begin{array}{rcl}
SYSTEM & \widehat{=} & (LIFT\ {}^{A \cup R}\!\parallel^{R \cup P}\ BUTTONS) \setminus R \\
LIFT & \widehat{=} & LIFT_0 \\
LIFT_0 & \widehat{=} & req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\uparrow} \\
 & & \Box req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\
 & & \Box req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\
LIFT_1^{\uparrow} & \widehat{=} & req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\
 & & \Box req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\
 & & \Box req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\uparrow} \\
LIFT_1^{\downarrow} & \widehat{=} & req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\
 & & \Box req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\
 & & \Box req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\downarrow}
\end{array}$$

$$
\begin{aligned}
LIFT_2 \;&\mathrel{\widehat{=}}\; req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\downarrow}\\
&\qquad \Box req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0\\
&\qquad \Box req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2\\
BUTTONS \;&\mathrel{\widehat{=}}\; BUTTON_0 \quad BUTTON_1 \quad BUTTON_2\\
BUTTON_i \;&\mathrel{\widehat{=}}\; push_i \xrightarrow{1} req_i \xrightarrow{1} BUTTON_i \qquad (i = 0, 1, 2)
\end{aligned}
$$

where the interleaving of the buttons could be either left- or right-biased, and

$$A \mathrel{\widehat{=}} \{arrive_i \mid i \in 0 \,..\, 2\} \qquad R \mathrel{\widehat{=}} \{req_i \mid i \in 0 \,..\, 2\} \qquad P \mathrel{\widehat{=}} \{push_i \mid i \in 0 \,..\, 2\}$$

$LIFT_0$ and $LIFT_2$ represent the lift on the ground and second floors respectively; $LIFT_1^{\uparrow}$ and $LIFT_1^{\downarrow}$ represent the lift on the first floor where the previous movement was up or down, respectively. The lift is biased in favour of next going to an adjacent floor; when it is on the first floor, it is biased in favour of continuing in the direction it last went.

We will show that the lift arrives within 15 seconds of the button being pressed if the environment always allows the *arrive* events.

$SYSTEM \; \mathsf{sat}_\rho \; SPEC$

where $SPEC \mathrel{\widehat{=}} \mathsf{internal}\, A \wedge push_i \;\mathsf{at}\; t \wedge \mathsf{beyond}\; t + 15 \Rightarrow arrive_i \;\mathsf{at}\; [t + 3, t + 15)$

Using the proof rule for hiding, we can reduce our proof obligation to

$$LIFT \;{}^{A\cup R}\|\!|\!|^{R\cup P}\; BUTTONS \;\mathsf{sat}_\rho$$
$$\mathsf{internal}\, A \wedge \mathsf{internal}\, R \Rightarrow \begin{pmatrix} push_i \;\mathsf{at}\; t \wedge \mathsf{beyond}\; t + 13 \Rightarrow req_i \;\mathsf{at}\; [t + 1, t + 13) \\ \wedge\; req_i \;\mathsf{at}\; t \wedge \mathsf{beyond}\; t + 2 \Rightarrow arrive_i \;\mathsf{at}\; t + 2 \end{pmatrix}$$

A req_i occurs within 13 seconds of a $push_i$, and $arrive_i$ occurs 2 seconds after the req_i.
We will use the proof rule for parallel composition to reduce the proof obligation to

$$
\begin{aligned}
LIFT \;&\mathsf{sat}_\rho\; SPEC_L\\
BUTTONS \;&\mathsf{sat}_\rho\; push_i \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; req_i \;\mathsf{at}\; [t, t + 1) \wedge req_i \;\mathsf{live\ from}\; t + 1
\end{aligned}
$$

where

$$
\begin{aligned}
SPEC_L \mathrel{\widehat{=}}\; & A, R \;\mathsf{separate}\\
& \wedge\; req_i \;\mathsf{at}\; t \Rightarrow arrive_i \;\mathsf{live\ from}\; t + 2\\
& \wedge\; \mathsf{internal}\, A \wedge req_i \;\mathsf{accessible\ from}\; t \wedge \mathsf{beyond}\; t + 12 \Rightarrow req_i \;\mathsf{at}\; [t, t + 12)
\end{aligned}
$$

The lift offers *req* and *arrive* events separately; two seconds after performing a req_i, it offers $arrive_i$; and if the environment is willing to perform any *arrive* events and is offering req_i, then the lift performs req_i within 12 seconds. The buttons offer req_i one second after performing $push_i$. We have the following proof obligation:

Lemma 5.5.1: Let $\sqsubseteq \mathrel{\widehat{=}} \sqsubseteq_L \;{}^{A\cup R}\|\!|\!|^{R\cup P}\; \sqsubseteq_B$. Then

$$
\begin{pmatrix} SPEC_L(\tau, \sqsubseteq_L, s \quad A \cup R) \\ \wedge\; (push_i \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; req \;\mathsf{at}\; [t, t + 1) \wedge req_i \;\mathsf{live\ from}\; t + 1)(\tau, \sqsubseteq_B, s \quad R \cup P) \end{pmatrix} \Rightarrow
$$
$$
\begin{pmatrix} \mathsf{internal}\, A \wedge \mathsf{internal}\, R \Rightarrow \\ \qquad \begin{pmatrix} push_i \;\mathsf{at}\; t \wedge \mathsf{beyond}\; t + 13 \Rightarrow req_i \;\mathsf{at}\; [t + 1, t + 13) \\ \wedge\; req_i \;\mathsf{at}\; t \wedge \mathsf{beyond}\; t + 2 \Rightarrow arrive_i \;\mathsf{at}\; t + 2 \end{pmatrix} \end{pmatrix} (\tau, \sqsubseteq, s)
$$

♡

Proof of lemma: Let $\sqsubseteq \mathrel{\widehat{=}} \sqsubseteq_L {}^{A\cup R}\|^{R \cup P} \sqsubseteq_B$. Suppose

$$SPEC_L(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$$
$$\wedge\ (push_i \text{ at } t \Rightarrow \text{no } req \text{ at } [t, t+1) \wedge req_i \text{ live from } t+1)(\tau, \sqsubseteq_B, s \upharpoonright R \cup P)$$
$$\wedge\ (\text{internal } A \wedge \text{internal } R)(\tau, \sqsubseteq, s)$$

We want to show

$$\begin{pmatrix} push_i \text{ at } t \wedge \text{beyond } t+13 \Rightarrow req_i \text{ at } [t+1, t+13) \\ \wedge\ req_i \text{ at } t \wedge \text{beyond } t+2 \Rightarrow arrive_i \text{ at } t+2 \end{pmatrix} (\tau, \sqsubseteq, s) \qquad (*)$$

For the first conjunct, suppose that $(push_i \text{ at } t \wedge \text{beyond } t+13)(\tau, \sqsubseteq, s)$. Then we have $(push_i \text{ at } t)(\tau, \sqsubseteq_B, s \upharpoonright R \cup P)$ so by $BUTTONS$' specification, $(req_i \text{ live from } t+1)(\tau, \sqsubseteq_B, s \upharpoonright R \cup P)$. We will show $(req_i \text{ accessible from } t+1)(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$. Recall corollary 5.3.2 which said:

> Let $\sqsubseteq = \sqsubseteq_P {}^{A}\|^{B} \sqsubseteq_Q$ and $c \in C \subseteq A \cap B$. Then if $(c \text{ live } I)(\tau, \sqsubseteq_Q, s \upharpoonright B)$ and $(\text{internal } C \wedge c, A \setminus C \text{ separate } I)(\tau, \sqsubseteq, s)$ then $(c \text{ accessible } I)(\tau, \sqsubseteq_P, s \upharpoonright A)$.

Taking $c = req_i$, $C = R$, we must show that $(\text{internal } R \wedge req_i, A \text{ separate } I)(\tau, \sqsubseteq, s)$. The first clause holds by hypothesis; the secoud holds because A, R separate. Hence we have

$$(req_i \text{ accessible from } t+1)(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$$

Also, $(\text{internal } A)(\tau, \sqsubseteq, s)$ so $(\text{internal } A)(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$. Hence from the third clause of $SPEC_L$ we see $(req_i \text{ at } [t+1, t+13))(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$, so $(req_i \text{ at } [t+1, t+13))(\tau, \sqsubseteq, s)$, as required.

For the second conjunct, of $(*)$, suppose that $(req_i \text{ at } t \wedge \text{beyond } t+2)$. Then $(req_i \text{ at } t)(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$ so $(arrive_i \text{ live from } t+2)(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$, by the second clause of $SPEC_L$. But $(\text{internal } A)(\tau, \sqsubseteq, s)$ so $(\text{internal } A)(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$, so $(arrive_i \text{ at } t+2)(\tau, \sqsubseteq_L, s \upharpoonright A \cup R)$, which gives $(arrive_i \text{ at } t+2)(\tau, \sqsubseteq, s)$, as required. □

We now prove that the two subcomponents satisfy their specifications. The result for the buttons is easily proved: we can use the proof rule for interleaving to reduce the proof obligation to

$$\forall i \bullet BUTTON_i \text{ sat}_\rho\ push_i \text{ at } t \Rightarrow \text{no } req \text{ at } [t, t+1) \wedge req_i \text{ live from } t+1$$

which can be proved using the proof rules for prefixing and recursion.

We show that $LIFT$ satisfies the specification $SPEC_L$ by proving that

$$\begin{array}{ll} LIFT_0 \ \mathbf{sat}_\rho\ SPEC_0 & LIFT_1^{\uparrow} \ \mathbf{sat}_\rho\ SPEC_1^{\uparrow} \\ LIFT_1^{\downarrow} \ \mathbf{sat}_\rho\ SPEC_1^{\downarrow} & LIFT_2 \ \mathbf{sat}_\rho\ SPEC_2 \end{array}$$

where

$$SPEC_0 \mathrel{\widehat{=}} SPEC_L$$
$$\wedge \begin{pmatrix} \text{internal } A \wedge \text{silent before } t \\ \wedge\ \text{beyond } t+9 \end{pmatrix} \Rightarrow \begin{pmatrix} req_0 \text{ accessible from } t \Rightarrow req_0 \text{ at } [t, t+9] \\ \wedge\ req_1 \text{ accessible from } t \Rightarrow req_1 \text{ at } t \\ \wedge\ req_2 \text{ accessible from } t \Rightarrow req_2 \text{ at } [t, t+3] \end{pmatrix}$$

$$SPEC_1^{\uparrow} \mathrel{\widehat{=}} SPEC_L$$
$$\wedge \begin{pmatrix}\text{internal } A \wedge \text{silent before } t \\ \wedge \text{ beyond } t+6\end{pmatrix} \Rightarrow \begin{pmatrix} req_0 \text{ accessible from } t \Rightarrow req_0 \text{ at } [t, t+6] \\ \wedge\ req_1 \text{ accessible from } t \Rightarrow req_1 \text{ at } [t, t+3] \\ \wedge\ req_2 \text{ accessible from } t \Rightarrow req_2 \text{ at } t \end{pmatrix}$$

$$SPEC_1^{\downarrow} \mathrel{\widehat{=}} SPEC_L$$
$$\wedge \begin{pmatrix}\text{internal } A \wedge \text{silent before } t \\ \wedge \text{ beyond } t+6\end{pmatrix} \Rightarrow \begin{pmatrix} req_0 \text{ accessible from } t \Rightarrow req_0 \text{ at } t \\ \wedge\ req_1 \text{ accessible from } t \Rightarrow req_1 \text{ at } [t, t+3] \\ \wedge\ req_2 \text{ accessible from } t \Rightarrow req_2 \text{ at } [t, t+6] \end{pmatrix}$$

$$SPEC_2 \mathrel{\widehat{=}} SPEC_L$$
$$\wedge \begin{pmatrix}\text{internal } A \wedge \text{silent before } t \\ \wedge \text{ beyond } t+9\end{pmatrix} \Rightarrow \begin{pmatrix} req_0 \text{ accessible from } t \Rightarrow req_0 \text{ at } [t, t+3] \\ \wedge\ req_1 \text{ accessible from } t \Rightarrow req_1 \text{ at } t \\ \wedge\ req_2 \text{ accessible from } t \Rightarrow req_2 \text{ at } [t, t+9] \end{pmatrix}$$

Note that $SPEC_0 \Rightarrow SPEC_L$ and $LIFT$ is defined to be $LIFT_0$, so this will be enough to deduce $LIFT$ $\mathbf{sat}_\rho$ $SPEC_L$.

We prove these results using the inference rule for mutual recursion, noting that the recursions are uniformly 3-constructive. We assume

$$\begin{array}{ll} LIFT_0 \ \mathbf{sat}_\rho \ SPEC_0 & LIFT_1^{\uparrow} \ \mathbf{sat}_\rho \ SPEC_1^{\uparrow} \\ LIFT_1^{\downarrow} \ \mathbf{sat}_\rho \ SPEC_1^{\downarrow} & LIFT_2 \ \mathbf{sat}_\rho \ SPEC_2 \end{array}$$

and we need to show

$$\begin{pmatrix} req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\uparrow} \\ \Box req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\ \Box req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \end{pmatrix} \ \mathbf{sat}_\rho \ SPEC_0$$

$$\begin{pmatrix} req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\ \Box req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\ \Box req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\uparrow} \end{pmatrix} \ \mathbf{sat}_\rho \ SPEC_1^{\uparrow}$$

$$\begin{pmatrix} req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\ \Box req_2 \xrightarrow{2} arrive_2 \xrightarrow{1} LIFT_2 \\ \Box req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\downarrow} \end{pmatrix} \ \mathbf{sat}_\rho \ SPEC_1^{\downarrow}$$

$$\begin{pmatrix} req_1 \xrightarrow{2} arrive_1 \xrightarrow{1} LIFT_1^{\downarrow} \\ \Box req_0 \xrightarrow{2} arrive_0 \xrightarrow{1} LIFT_0 \\ \Box req_2 \xrightarrow{3} arrive_2 \xrightarrow{1} LIFT_2 \end{pmatrix} \ \mathbf{sat}_\rho \ SPEC_2$$

We prove the first result; the rest are similar.

We begin by proving that $SPEC_L$ is satisfied. For the separate clause it is enough, by the proof rule for external choice, to show that

$$req_j \xrightarrow{2} arrive_j \xrightarrow{1} LIFT_j^* \ \mathbf{sat}_\rho \ A, R \text{ separate}$$

where $LIFT_0^* \mathrel{\widehat{=}} LIFT_0$, $LIFT_1^* \mathrel{\widehat{=}} LIFT_1^{\uparrow}$, etc. This is easily proved using the proof rule for prefixing and the assumption that $LIFT_j^*$ $\mathbf{sat}_\rho$ A, R separate.

For the second clause of $SPEC_L$ it is enough, again by the proof rule for external choice, to show that

$$req_j \xrightarrow{2} arrive_j \xrightarrow{1} LIFT_j^* \ \mathbf{sat}_\rho \ req_i \text{ at } t \Rightarrow arrive_i \text{ live from } t+2$$

Suppose $(req_i \text{ at } t)(\tau, \sqsubseteq, s)$. Then either

- this is the first event of $req_j \xrightarrow{2} arrive_j \xrightarrow{1} LIFT_j^*$, in which case $i = j$. Then using the proof rule for prefixing, we have $(arrive_j \text{ live from } t+2)(\tau, \sqsubseteq, s)$; *or*
- this is not the first event, in which case it must be an event of $LIFT_j^*$. In this case, we can deduce the result from the corresponding clause of the assumption about $LIFT_j^*$.

For the third clause of $SPEC_L$, suppose

$$(\text{internal } A \wedge req_i \text{ accessible from } t \wedge \text{beyond } t+12)(\tau, \sqsubseteq, s)$$

We want to show req_i at $[t, t+12)$. Note that $(t', req_1) \sqsupset (t', req_2) \sqsupset (t', req_0) \sqsupset (t', \{\!|\!\})$ for all times t' up until when the first event occurs. Expanding the definition of req_i accessible from t, we see that an event must occur by time t at the latest. We have a number of cases to consider.

- Suppose no event occurs before time t, and req_i occurs at t: in this case the result is immediate.
- Suppose the first event to occur is req_j at time t, and $j \neq i$: by the definition of $\boxdot$, and since req_i accessible from t we must have $j = 2$ and $i = 0$ or $j = 1$ and $i \neq 1$. We consider these two subcases:
  - Case $j = 2$ and $i = 0$: because $(\text{internal } A)(\tau, \sqsubseteq, s)$, by assumption, and by the rule for prefixing, we have $arrive_2$ at $t+2$, and the process acts like $LIFT_2$ from $t+3$. Now by assumption.

    $LIFT_2$ $\mathbf{sat}_\rho$

    $$\text{internal } A \wedge \text{silent before } 0 \wedge \text{beyond } 9 \wedge req_0 \text{ accessible from } 0 \Rightarrow req_0 \text{ at } [0, 3]$$

    so in this case we have req_0 at $[t+3, t+6]$.
  - Case $j = 1$ and $i \neq 1$: because of the assumption $(\text{internal } A)(\tau, \sqsubseteq, s)$. and by the rule for prefixing, we have $arrive_1$ at $t+2$, and the process acts like $LIFT_1^{\uparrow}$ from $t+3$. Now by assumption,

    $$LIFT_1^{\uparrow} \ \mathbf{sat}_\rho \ (\text{internal } A \wedge \text{silent before } 0 \wedge \text{beyond } 9) \Rightarrow \begin{pmatrix} req_0 \text{ accessible from } 0 \Rightarrow req_0 \text{ at } [0, 6] \\ \wedge\ req_2 \text{ accessible from } 0 \Rightarrow req_2 \text{ at } 0 \end{pmatrix}$$

    so in either case we have req_i at $[t+3, t+9]$.

- Suppose the first event to occur is req_j at some time t' with $t' < t < t' + 3$: then we have $arrive_j$ at $t' + 2$ and the process acts like $LIFT_j^*$ from time $t' + 3$. Now by assumption,

$$LIFT_j^* \ \mathbf{sat}_\rho \begin{pmatrix} \mathsf{internal}\ A \wedge \mathsf{silent\ before}\ 0 \\ \wedge\ \mathsf{beyond}\ 9 \wedge req_i\ \mathsf{accessible\ from}\ 0 \end{pmatrix} \Rightarrow req_i\ \mathsf{at}\ [0, 9]$$

  so req_i at $[t' + 3, t' + 12]$, i.e. req_i at $(t, t + 12)$.

- Suppose the first event to occur is req_j at some time t' with $t' + 3 \quad t$: then we have $arrive_j$ at $t' + 2$ and the process acts like $LIFT_j^*$ from time $t' + 3$. Now by assumption,

$$LIFT_j^* \ \mathbf{sat}_\rho\ \mathsf{internal}\ A \wedge req_i\ \mathsf{accessible\ from}\ t \wedge \mathsf{beyond}\ t + 12 \Rightarrow req_i\ \mathsf{at}\ [t, t + 12)$$

  so req_i at $[t, t + 12)$.

Hence in each case we have req_i at $[t, t + 12)$, so the second clause of $SPEC_L$ is satisfied.

We now turn our attention to proving

$$\mathsf{internal}\ A \wedge \mathsf{silent\ before}\ t \wedge \mathsf{beyond}\ t + 9 \Rightarrow \begin{pmatrix} req_0\ \mathsf{accessible\ from}\ t \Rightarrow req_0\ \mathsf{at}\ [t, t + 9] \\ \wedge\ req_1\ \mathsf{accessible\ from}\ t \Rightarrow req_1\ \mathsf{at}\ t \\ \wedge\ req_2\ \mathsf{accessible\ from}\ t \Rightarrow req_2\ \mathsf{at}\ [t, t + 3] \end{pmatrix}$$

Assume internal A $\wedge$ silent before t $\wedge$ beyond $t + 9$. We prove the first clause of the consequent; the other clauses are easier. Suppose then that (req_0 accessible from t)$(\tau, \sqsubseteq, s)$. By the definition of $\sqsubseteq$ we have $(t, req_1) \sqsupset (t, req_2) \sqsupset (t, req_0) \sqsupset (t, \{\!|\,|\!\})$, and expanding the definition of req_0 accessible from t we have $(t, req_0) \not\sqsupset s \uparrow t$, so a req_i occurs at t. We consider the three possibilities:

- Case req_0 at t: the result is immediate.

- Case req_1 at t: then $arrive_1$ at $t + 2$, and the process acts like $LIFT_1^\dagger$ from time $t + 3$. Now by assumption

$$LIFT_1^\dagger \ \mathbf{sat}_\rho \begin{pmatrix} \mathsf{internal}\ A \wedge \mathsf{silent\ before}\ 0 \\ \wedge\ \mathsf{beyond}\ t + 9 \wedge req_0\ \mathsf{accessible\ from}\ 0 \end{pmatrix} \Rightarrow req_0\ \mathsf{at}\ [0, 6]$$

  so req_0 at $[t + 3, t + 9]$.

- Case req_2 at t: this case is similar to the previous case.

So the result holds in each case.

Hence we have shown that $LIFT\ \mathbf{sat}_\rho\ SPEC_L$ and so $SYSTEM\ \mathbf{sat}_\rho\ SPEC$. $\square$

Chapter 6

Relating the Prioritized Model to the Timed Failures Model

In this chapter we want to relate the Prioritized Model of BTCSP to the Timed Failures Model of Timed CSP. This will help us to understand the Prioritized Model, and also allow us to prove properties of prioritized processes by proving results about corresponding unprioritized processes.

In section 6.1 we produce the abstraction mapping from the Prioritized Model of BTCSP to the Timed Failures Model of Timed CSP. We present a syntactic mapping φ that removes all priorities, and derive a corresponding semantic mapping θ. We show that under the abstraction mapping θ, the set of failures corresponding to a prioritized process P is a subset of the failures of the process $\varphi(P)$.

In section 6.2 we use this abstraction result to show how, under certain circumstances, we can translate specifications in the Prioritized Model into corresponding specifications in the Timed Failures Model. If we can find a TCSP process that satisfies a failures specification, then any BTCSP refinement of that process will satisfy a corresponding specification in the Prioritized Model. We develop a number of rules for translating specifications written in our specification language. The Timed Failures Model is a simpler model than the Prioritized Model of BTCSP, so the proofs are normally simpler.

We can also use this refinement method as follows: often a specification will consists of a number of conjuncts; normally it is possible to find a failures specification corresponding to *most* of these conjuncts. If we can find a TCSP process satisfying this failures specification, then we only need to investigate which of its BTCSP refinements satisfy the rest of the conjuncts of the original specification. We illustrate this method with an example in section 6.3.

6.1 An abstraction result

In chapter 5 we presented abstraction mappings from the probabilistic and deterministic languages and models to the prioritized language BTCSP and model $\mathcal{M}_{TB}$. We now present abstraction mappings from BTCSP to TCSP, and from the Biased Model $\mathcal{M}_{TB}$ to the Timed Failures Model $\mathcal{M}_{TF}$. The abstraction mappings are shown in figure 6.1. The mappings $\varphi_P^{(B)}$ and $\theta_P^{(B)}$ remove *p*robabilities while keeping *b*iases; the mappings $\varphi_D^{(B)}$ and $\theta_D^{(B)}$ remove *d*eterminism but keep *b*iases; the mappings φ_B and θ_B remove *b*iases.

$$\begin{tikzcd}[column sep=small] PBTCSP \arrow[dr, "\varphi_P^{(B)}"'] & & DTCSP \arrow[dl, "\varphi_D^{(B)}"] \\ & BTCSP \arrow[d, "\varphi_B"'] & \\ & TCSP & \end{tikzcd} \qquad \begin{tikzcd}[column sep=small] \mathcal{M}_{PTB} \arrow[dr, "\theta_P^{(B)}"'] & & \mathcal{M}_{DTB} \arrow[dl, "\theta_D^{(B)}"] \\ & \mathcal{M}_{TB} \arrow[d, "\theta_B"'] & \\ & \mathcal{M}_{TF} & \end{tikzcd}$$

Figure 6.1: A hierarchy of languages and models

We define the obvious mapping from the syntax of BTCSP to the syntax of TCSP which removes all priorities:

Definition 6.1.1: We define $\varphi_B : BTCSP \to TCSP$ by

$$\begin{array}{rcl} \varphi_B(P \sqsubset\!\!\!\square\, Q) \mathrel{\hat=} \varphi_B(P) \quad \varphi_B(Q) & & \varphi_B(P \,\square\!\!\!\sqsupset Q) \mathrel{\hat=} \varphi_B(P) \quad \varphi_B(Q) \\ \varphi_B(P \,{}^X\!\!\Vdash^Y Q) \mathrel{\hat=} \varphi_B(P) \,{}^X\!\|^Y \varphi_B(Q) & & \varphi_B(P \,{}^X\!\!\Vdash^Y Q) \mathrel{\hat=} \varphi_B(P) \,{}^X\!\|^Y \varphi_B(Q) \\ \varphi_B(P \Vdash Q) \mathrel{\hat=} \varphi_B(P) \,\|\, \varphi_B(Q) & & \varphi_B(P \Vdash Q) \mathrel{\hat=} \varphi_B(P) \,\|\, \varphi_B(Q) \\ \varphi_B(P \longleftarrow Q) \mathrel{\hat=} \varphi_B(P) \quad \varphi_B(Q) & & \varphi_B(P \longrightarrow Q) \mathrel{\hat=} \varphi_B(P) \quad \varphi_B(Q) \\ \varphi_B(P \underset{C}{\Vdash} Q) \mathrel{\hat=} \varphi_B(P) \underset{C}{\|} \varphi_B(Q) & & \varphi_B(P \underset{C}{\Vdash} Q) \mathrel{\hat=} \varphi_B(P) \underset{C}{\|} \varphi_B(Q) \\ \varphi_B(P) \mathrel{\hat=} P & & \text{for } P = STOP, SKIP, WAIT\ t, \text{ or } X \\ \varphi_B(F(P)) \mathrel{\hat=} F(\varphi_B(P)) & & \text{for } F(P) = a \xrightarrow{t} P, WAIT\ t\,; P, P \setminus X, \\ & & \quad f(P), \mu\, X \quad P, \text{ or } \mu\, X \quad P \\ \varphi_B(P \oplus Q) \mathrel{\hat=} \varphi_B(P) \oplus \varphi_B(Q) & & \text{for } \oplus = \sqcap, \ , \ \overset{t}{}, \ {}_t, \text{ or } \underset{e}{\nabla} \end{array}$$

◇

6.1.1 The abstraction mapping

Having produced a mapping between the syntaxes, we now seek a corresponding mapping θ_B between the semantic spaces so that the diagram in figure 6.2 commutes.

We might naïvely expect to be able to produce a result of the following form:

$$\text{if } \forall X \quad \theta_B(\rho\ X\ ) = \rho'\ X \quad \text{then } \theta_B(\mathcal{A}_{BT}\ P\ \rho) = \mathcal{F}_T\ \varphi_B(P)\ \rho'$$

However, it is not possible to produce such a mapping θ_B. Consider the two processes

$$P \mathrel{\hat=} a \longrightarrow b \longrightarrow STOP \sqsubset\!\!\!\square\, a \longrightarrow c \longrightarrow STOP \qquad \text{and} \qquad Q \mathrel{\hat=} a \longrightarrow b \longrightarrow STOP$$

$$
\begin{tikzcd}
BTCSP \arrow[r, "\varphi_B"] \arrow[d, "\mathcal{A}_{BT}"'] & TCSP \arrow[d, "\mathcal{F}_T"] \\
\mathcal{M}_{TB} \arrow[r, "\theta_B"'] & \mathcal{M}_{TF}
\end{tikzcd}
$$

Figure 6.2: The syntactic and semantic maps

In $\mathcal{M}_{TB}$ these two processes are equivalent, so we have $\theta_B(\mathcal{A}_{BT}\ P\ \rho) = \theta_B(\mathcal{A}_{BT}\ Q\ \rho)$, whereas $\mathcal{F}_T\ \varphi_B(P)\ \rho' \neq \mathcal{F}_T\ \varphi_B(Q)\ \rho'$ (for any environments ρ and ρ'). We shall give a function θ_B such that for all BTCSP processes P,

$$\text{if } \forall X \quad \theta_B(\rho\ X\ ) = \rho'\ X \quad \text{then } \theta_B(\mathcal{A}_{BT}\ P\ \rho) \subseteq \mathcal{F}_T\ \varphi_B(P)\ \rho'$$

To begin with, we want to be able to convert between traces in the Prioritized Model and traces in the Failures Model. In the Prioritized Model, traces are represented as functions from an initial segment of time to actions: $TT \mathrel{\widehat{=}} \{s : TIME \nrightarrow ACT \mid \exists\,\tau \quad \mathrm{dom}\, s = [0, \tau]\}$. In the Failures Model they are represented as sequences of timed events: $T\Sigma^*_{\leqslant}$. We therefore require the following definition.

Definition 6.1.2: We define a relation $\_ \sim \_ : T\Sigma^*_{\leqslant} \times TT$ by

$$
\begin{aligned}
\langle\rangle &\sim \lambda\, t \quad \{\!|\,|\!\} \\
(t, a) \quad s &\sim s' \oplus \{t \mapsto s'(t) \uplus \{\!|a|\!\}\}, \text{ if } s \sim s'
\end{aligned}
$$

$\diamond$

Informally, $s \sim s'$ if s and s' represent the same trace.

An event (t, a) will be refused during a behaviour $(\tau, \sqsubseteq, s)$ if the process would rather not perform an extra a: in other words, if the process prefers what it performs (i.e. $s \uparrow t$) to (t, a) added to what it performs (i.e. $s \uparrow t \uplus (t, a)$). Formally, this condition can be expressed as $s \uparrow t \uplus (t, a) \not\sqsupseteq s \uparrow t$.

Note that if $s \uparrow t \uplus (t, a) \not\sqsupseteq s \uparrow t$ and $s \uparrow t \uplus (t, b) \not\sqsupseteq s \uparrow t$ then $s \uparrow t \uplus (t, \{\!|a, b|\!\}) \not\sqsupseteq s \uparrow t$, as can be easily verified from axiom A6. In other words, if a process can refuse an a, and it can refuse a b, then it can refuse the a and the b together.

However, it turns out that it is not enough to take the set $\{(t, a) \mid t < \tau \wedge s \uparrow t \uplus (t, a) \not\sqsupseteq s \uparrow t\}$ as the total refusal of the behaviour $(\tau, \sqsubseteq, s)$. Consider the behaviour

$$(1,\ [0, 0] \otimes \langle\{\!|a|\!\}, \{\!|\,|\!\}\rangle \quad (0, 1] \otimes \langle\{\!|\,|\!\}\rangle,\ \prec\succ)$$

of $(a \mathrel{\square} b) \setminus b$, where a b is performed silently at time 0. For this behaviour, if we put $\aleph \mathrel{\widehat{=}} \{(t, a) \mid t < \tau \wedge s \uparrow t \uplus (t, a) \not\sqsupseteq s \uparrow t\}$, then we have $(0, 1) \times \{a\} \subseteq \aleph$, but $(0, a) \notin \aleph$ contrary to our expectations. To fit in with the Timed Failures Model, we will require that the total refusal relating to a behaviour is closed on the left; this means that the total refusal for the above behaviour will include $[0, 1) \times \{a\}$.

We can now define a function giving the total refusal relating to a behaviour.

Definition 6.1.3: The total refusal of a behaviour $(\tau, \sqsubseteq, s)$ is given by $ref(\tau, \sqsubseteq, s)$ where the function $ref : BEH \to RSET$ is defined by

$$ref(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} \text{closure}\{(t, a) \mid t < \tau \wedge s \uparrow t \uplus (t, a) \not\sqsupset s \uparrow t\}$$

where closure S is the left-hand closure of S

$$\text{closure}\, S = \{(t, a) \mid \exists \varepsilon > 0 \quad (t, t + \varepsilon) \times \{a\} \subseteq S\}$$

$\diamondsuit$

The following results about the total refusal of a process will prove useful. The total refusal is open on the right in the sense that if the timed event (t, a) is refused, then (t', a) is refused for all times t' 'just after' t.

Lemma 6.1.4: $(t, a) \in ref(\tau, \sqsubseteq, s) \Rightarrow \exists \varepsilon > 0 \quad [t, t + \varepsilon) \times \{a\} \subseteq ref(\tau, \sqsubseteq, s)$ $\heartsuit$

Proof: Suppose $(t, a) \in ref(\tau, \sqsubseteq, s)$ and suppose for a contradiction that the consequence of the lemma does not hold. Then by the definition of $ref(\tau, \sqsubseteq, s)$ and the finite variability condition on offer relations (axiom A8), we must have for some $\varepsilon > 0$ that $\forall t' \in (t, t + \varepsilon)$ $s \uparrow t' \uplus (t', a) \sqsupset s \uparrow t'$. Hence by the sub-bag closure condition on offers (condition A5), $\forall t' \in (t, t + \varepsilon) \quad (t', a) \in \text{items} \sqsubseteq$. Then by condition A3 we have that $s \uparrow t \uplus (t, a) \sqsupset s \uparrow t$, contradicting our assumption that $(t, a) \in ref(\tau, \sqsubseteq, s)$. $\square$

We use this result to prove that $ref(\tau, \sqsubseteq, s)$ is a member of the set $RSET$ of refusals.

Lemma 6.1.5: $ref(\tau, \sqsubseteq, s) \in RSET$. $\heartsuit$

Proof: $ref(\tau, \sqsubseteq, s)$ is closed on the left by definition, open on the right by the previous lemma, and satisfies the finite variability condition by the corresponding condition on offer relations (axiom A8). $\square$

We claim that a timed failure $(s', \aleph)$ could have resulted from a prioritized behaviour $(\tau, \sqsubseteq, s)$ precisely when $s' \sim s \wedge \aleph \subseteq ref(\tau, \sqsubseteq, s)$. We will write $(s', \aleph) \simeq (\tau, \sqsubseteq, s)$ and say $(s', \aleph)$ is compatible with $(\tau, \sqsubseteq, s)$ if this holds.

Definition 6.1.6: For all $(s', \aleph) \in TF$ and $(\tau, \sqsubseteq, s) \in BEH$,

$$(s', \aleph) \simeq (\tau, \sqsubseteq, s) \Leftrightarrow s' \sim s \wedge \aleph \subseteq ref(\tau, \sqsubseteq, s)$$

$\diamondsuit$

If $(s', \aleph) \simeq (\tau, \sqsubseteq, s)$ then $(s', \aleph)$ is compatible with $(\tau, \sqsubseteq, s)$, in the sense that

- s and s' represent the same trace, i.e. $s' \sim s$; *and*
- all the members of $\aleph$ are refusals of the behaviour $(\tau, \sqsubseteq, s)$, i.e. $\aleph \subseteq ref(\tau, \sqsubseteq, s)$.

We can now give the mapping between our semantic spaces.

Definition 6.1.7: The function $\theta_B : \mathcal{S}_{TB} \rightarrow \mathcal{S}_{TF}$ is given by

$$\theta_B(A) \mathrel{\hat=} \{(s', \aleph) : TF \mid \exists (\tau, \sqsubseteq, s) \in A \quad (s', \aleph) \simeq (\tau, \sqsubseteq, s)\}$$

$\diamond$

$\theta_B(A)$ is the set of all timed failures that are compatible with some member of A.
Recall our definition of failures environments:

$$ENV_F \mathrel{\hat=} VAR \rightarrow \mathcal{S}_{TF}$$

We write σ for a typical member of ENV_F, and $\sigma \ X$ for the set of failures associated with variable X. The priorities environment ρ and failures environment σ are compatible, in the sense that they associate the same processes with each variable, if $\forall X : VAR \quad \theta_B(\rho \ X\ ) = \sigma \ X$; this can be written more concisely as $\sigma = \theta_B \circ \rho$.
The composition of θ_B with $\mathcal{A}_{BT}$ will be sufficiently important that we give it a name:

Definition 6.1.8: The function $\mathcal{A}_{FT} : BTCSP \rightarrow ENV_F \rightarrow \mathcal{S}_{TF}$ is given by

$$\mathcal{A}_{FT} \ P \ \sigma \mathrel{\hat=} \theta_B(\mathcal{A}_{BT} \ P \ \rho) \quad \text{where} \quad \sigma = \theta_B \circ \rho$$

$\diamond$

Note that although there may in general be several environments ρ satisfying the condition that $\sigma = \theta_B \circ \rho$, this definition is independent of which one we choose: the only place where ρ is used is when giving a semantics to a variable; in this case we have

$$\mathcal{A}_{FT} \ X \ \sigma = \theta_B(\mathcal{A}_{BT} \ X \ \rho) = \theta_B(\rho \ X\ ) = \sigma \ X$$

so the choice of ρ makes no difference.
In the following subsections we will study the image of $\mathcal{M}_{TB}$ under the mapping θ_B and show that it is contained within the Failures Model $\mathcal{M}_{TF}$. We will then consider the effect of the mapping $\mathcal{A}_{FT}$ on the syntax of BTCSP.

6.1.2 The space $\theta(\mathcal{M}_{TB})$

All members of $\theta(\mathcal{M}_{TB})$ satisfy the healthiness conditions of $\mathcal{M}_{TF}$.

Theorem 6.1.9: For all S in $\theta(\mathcal{M}_{TB})$

1. $(\langle\rangle, \{\}) \in S$

2. $(s\ \ w, \aleph) \in S \Rightarrow (s, \aleph\ \ begin\, w) \in S$

3. $(s, \aleph) \in S \wedge s \cong w \Rightarrow (w, \aleph) \in S$

4. $(s, \aleph) \in S \wedge t\ \ \ 0 \Rightarrow$
$$\exists \aleph' \in RSET\ \ \aleph \subseteq \aleph' \wedge (s, \aleph') \in S$$
$$\wedge (t'\ \ t \wedge (t', a) \notin \aleph' \Rightarrow (s\ \ t'\ \ (t', a), \aleph'\ \ t') \in S)$$

5. $\forall t \in [0, \infty)\ \ \exists n(t) \in\ \ \ \forall (s, \aleph) \in S\ \ end\, s\ \ \ t \Rightarrow \# s\ \ \ n(t)$

6. $(s, \aleph) \in S \wedge \aleph' \in RSET \wedge \aleph' \subseteq \aleph \Rightarrow (s, \aleph') \in S$

7. $$\begin{pmatrix} (s\ \ w, \aleph) \in S \wedge \aleph' \in RSET \\ \wedge\ end\, s\ \ \ begin\, \aleph' \wedge end\, \aleph'\ \ \ begin\, w \\ \wedge \forall (t, a) \in \aleph'\ \ (s\ \ (t, a), \aleph\ \ t) \notin S \end{pmatrix} \Rightarrow (s\ \ w, \aleph \cup \aleph') \in S$$

♡

Proof: We prove each result in turn. Let $S = \theta_B(A)$.

1. Axiom B4 of $\mathcal{M}_{TB}$ states that there is some offer relation $\sqsubseteq$ such that $(0, \sqsubseteq, \prec\succ) \in A$. Then $\langle\rangle \sim \prec\succ$ and $\{\} \subseteq ref(0, \sqsubseteq, \prec\succ)$ so $(\langle\rangle, \{\}) \in S$.
2. If $(s\ \ w, \aleph) \in S$ then for some s', w', τ, $\sqsubseteq$ we have $s \sim s'$, $w \sim w'$, $\aleph \subseteq ref(\tau, \sqsubseteq, s'\ \ w')$ and $(\tau, \sqsubseteq, s'\ \ w') \in A$. Then by axiom B3, $(begin\, w, \sqsubseteq\ \ begin\, w, s') \in A$. Also $\aleph$ $begin\, w \subseteq ref(begin\, w, \sqsubseteq\ \ begin\, w, s')$ and so $(s, \aleph\ \ begin\, w) \in S$.
3. It is sufficient to show that if $s \sim s'$ and $s \cong w$ then $w \sim s'$, which follows directly from the definition of $\sim$.
4. Suppose $(s, \aleph) \in S \wedge t\ \ \ 0$. Then there is some $(\tau, \sqsubseteq, s') \in A$ such that $(s, \aleph) \simeq (\tau, \sqsubseteq, s')$. Let $\tau' \mathrel{\hat=} \tau \sqcup t$. Then $\exists \sqsubseteq'\ \ \sqsubseteq'\ \ \tau = \sqsubseteq \wedge (\tau', \sqsubseteq', s'\ \ \prec\succ) \in A$ by axiom B5. Let $\aleph' \mathrel{\hat=} ref(\tau', \sqsubseteq', s'\ \ \prec\succ)$. Then $\aleph' \in RSET$ by lemma 6.1.5; $\aleph \subseteq \aleph'$ by construction; $(s, \aleph') \in S$ by definition of θ_B; and if $t'\ \ \ t \wedge (t', a) \notin \aleph$ then $s' \uparrow t \uplus (t', a) \sqsupseteq' s' \uparrow t$ by definition of ref, so $s' \uparrow t \uplus (t', a) \in$ items $\sqsubseteq'$ and so $(t', \sqsubseteq'\ \ t', s'\ \ t'\ \ (t', a)) \in A$ by axiom B3, and hence $(s\ \ t'\ \ (t', a), \aleph\ \ t') \in S$ since $(s\ \ t'\ \ (t', a), \aleph\ \ t') \simeq (t', \sqsubseteq'\ \ t', s'$ $t'\ \ (t', a))$.
5. This follows directly from axiom B1.
6. Suppose $(s, \aleph) \in S \wedge \aleph' \in RSET \wedge \aleph' \subseteq \aleph$. Then $\exists (\tau, \sqsubseteq, s') \in A\ \ \ (s, \aleph) \simeq (\tau, \sqsubseteq, s')$. Hence $\aleph' \subseteq ref(\tau, \sqsubseteq, s')$ so $(s, \aleph') \simeq (\tau, \sqsubseteq, s')$ and so $(s, \aleph') \in S$.
7. Suppose the antecedents hold. Then $\exists (\tau, \sqsubseteq, s'\ \ w') \in A\ \ \ s \sim s' \wedge w \sim w' \wedge \aleph \subseteq$ $ref(\tau, \sqsubseteq, s'\ \ w')$. Then $\forall (t, a) \in \aleph'\ \ \ (t, \sqsubseteq\ \ t, s\ \ (t, a)) \notin A$ so $s \uparrow t \uplus (t, a) \notin$ items $\sqsubseteq$, from axiom B3, so $\aleph' \subseteq ref(\tau, \sqsubseteq, s'\ \ w')$. Hence $(s\ \ w, \aleph \cup \aleph') \in S$.

□

Hence $\theta(\mathcal{M}_{TB})$ lies within $\mathcal{M}_{TF}$.

Recall that the metric on $\mathcal{M}_{TF}$ is defined by

$d_F(S_P, S_Q) \mathrel{\widehat{=}} \inf(\{2^{-t} \mid S_P \quad t = S_Q \quad t\} \cup \{1\}) \qquad \text{where} \qquad S \quad t \mathrel{\widehat{=}} \{(s, \aleph) \in S \mid end(s, \aleph) \quad t\}$

We state a series of lemmas concerning this metric. If two processes "agree" up until some time in the Prioritized Model, then they "agree" up until that time in the Failures Model.

Lemma 6.1.10: If $A_P \quad t = A_Q \quad t$ then $\theta_B(A_P) \quad t = \theta_B(A_Q) \quad t$. ♡

Proof: This follows immediately from the fact that the failures of a process up to some time t depend only upon the prioritized behaviours up to time t, i.e. $(\theta_B(A)) \quad t = \theta_B(A \quad t)$. □

Processes are "closer" under the failures metric than under the priorities metric.

Lemma 6.1.11: For all sets A_P and A_Q of prioritized behaviours,

$$d_F(\theta_B(A_P), \theta_B(A_Q)) \quad d_B(A_P, A_Q)$$

where d_B is the metric in $\mathcal{M}_{TB}$. ♡

Proof: This follows immediately from the previous lemma and the definition of the metrics. □

Lemma 6.1.12: The mapping θ_B is continuous with respect to the metrics d_F and d_B. ♡

Proof: Suppose $\langle X_i \mid i \in \quad \rangle$ has limit X in $\mathcal{M}_{TB}$. Then we claim that $\langle \theta_B(X_i) \mid i \in \quad \rangle$ has limit $\theta_B(X)$. Pick $\varepsilon > 0$; then there is some N such that $\forall\, i > N \quad d_B(X_i, X) < \varepsilon$, so by the previous lemma, $\forall\, i > N \quad d_F(\theta_B(X_i), \theta_B(X)) \quad d_B(X_i, X) < \varepsilon$. □

6.1.3 The mapping $\mathcal{A}_{FT}$

The following theorem describes the effect of $\mathcal{A}_{FT}$ on the syntax of BTCSP.

Theorem 6.1.13: The function $\mathcal{A}_{FT}$ satisfies the following properties:

$\mathcal{A}_{FT} \ \ STOP\ \sigma = \{(\langle\rangle, \aleph) \mid \aleph \in RSET\}$

$\mathcal{A}_{FT} \ \ WAIT\ t\ \sigma = \{(\langle\rangle, \aleph) \mid \quad \notin \Sigma(\aleph \quad t)\} \cup \{((\langle(t', \quad)\rangle, \aleph) \mid t' \quad t \wedge \quad \notin \Sigma(\aleph \uparrow [t, t'))\}$

$\mathcal{A}_{FT} \ \ SKIP\ \sigma = \{(\langle\rangle, \aleph) \mid \quad \notin \Sigma\aleph\} \cup \{((\langle(t, \quad)\rangle, \aleph) \mid \quad \notin \Sigma(\aleph \quad t)\}$

$\mathcal{A}_{FT} \ \ X\ \sigma = \sigma\ X$

$$\mathcal{A}_{FT}\ a \xrightarrow{0} P\ \sigma =$$
$$\quad \{(\langle\rangle, \aleph) \mid a \notin \Sigma\aleph\} \cup \{(\langle(t,a)\rangle \quad s_P + t, \aleph) \mid a \notin \Sigma(\aleph \quad t) \wedge (s_P, \aleph - t) \in \mathcal{A}_{FT}\ P\ \sigma\}$$

$$\mathcal{A}_{FT}\ a \xrightarrow{t} P\ \sigma =$$
$$\quad \{(\langle\rangle, \aleph) \mid a \notin \Sigma\aleph\} \cup \{(\langle(t',a)\rangle \quad s_P + t + t', \aleph) \mid a \notin \Sigma(\aleph \quad t') \wedge (s_P, \aleph - t - t') \in \mathcal{A}_{FT}\ P\ \sigma\}$$

$$\mathcal{A}_{FT}\ P \quad Q\ \sigma \subseteq$$
$$\quad \{(s,\aleph) \mid \quad \notin \Sigma s \wedge \forall I \in \mathit{TINT} \quad (s, \aleph \cup I \times \{\quad\}) \in \mathcal{A}_{FT}\ P\ \sigma\}$$
$$\quad \cup\{(s,\aleph) \mid \exists t \quad \notin \Sigma(s \quad t) \wedge (s \quad t \quad \langle(t, \quad)\rangle, \aleph \quad t \cup [0,t) \times \{\quad\}) \in \mathcal{A}_{FT}\ P\ \sigma$$
$$\qquad \wedge\ s \uparrow (t, t+\delta) = \langle\rangle \wedge (s - t - \delta, \aleph - t - \delta) \in \mathcal{A}_{FT}\ Q\ \sigma\}$$

$$\mathcal{A}_{FT}\ \mathit{WAIT}\ t\,;P\ \sigma = \{(s+t, \aleph) \mid (s, \aleph - t) \in \mathcal{A}_{FT}\ P\ \sigma\}$$

$$\mathcal{A}_{FT}\ P \sqcap Q\ \sigma = \mathcal{A}_{FT}\ P\ \sigma \cup \mathcal{A}_{FT}\ Q\ \sigma$$

$$\mathcal{A}_{FT}\ \quad_{i \in I} P_i\ \sigma = \bigcup\{\mathcal{A}_{FT}\ P_i\ \sigma \mid i \in I\}$$

$$\mathcal{A}_{FT}\ P \,\Box\, Q\ \sigma, \mathcal{A}_{FT}\ P \,\Box\, Q\ \sigma \subseteq$$
$$\quad \{(\langle\rangle, \aleph) \mid (\langle\rangle, \aleph) \in \mathcal{A}_{FT}\ P\ \sigma \cap \mathcal{A}_{FT}\ Q\ \sigma\}$$
$$\quad \cup\{(s,\aleph) \mid s \neq \langle\rangle \wedge (s,\aleph) \in \mathcal{A}_{FT}\ P\ \sigma \cup \mathcal{A}_{FT}\ Q\ \sigma$$
$$\qquad \wedge (\langle\rangle, \aleph \quad \mathit{begin}\ s) \in \mathcal{A}_{FT}\ P\ \sigma \cap \mathcal{A}_{FT}\ Q\ \sigma\}$$

$$\mathcal{A}_{FT}\ c?a : A \xrightarrow{t_a} P_a\ \sigma =$$
$$\quad \{(\langle\rangle, \aleph) \mid c.A \cap \Sigma\aleph = \{\}\}$$
$$\quad \cup\{(\langle(t, c?a)\rangle \quad s + t + t_a, \aleph) \mid a \in A \wedge c.A \cap \Sigma(\aleph \quad t) = \{\} \wedge (s, \aleph - t - t_a) \in \mathcal{A}_{FT}\ P_a\ \sigma\}$$

$$\mathcal{A}_{FT}\ P \,|||\, Q\ \sigma, \mathcal{A}_{FT}\ P \,|||\, Q\ \sigma \subseteq \{(s, \aleph_P \cup \aleph_Q) \mid (s, \aleph_P) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s, \aleph_Q) \in \mathcal{A}_{FT}\ Q\ \sigma\}$$

$$\mathcal{A}_{FT}\ P\ {}_X|||_Y\ Q\ \sigma, \mathcal{A}_{FT}\ P\ {}_X|||_Y\ Q\ \sigma \subseteq$$
$$\quad \{(s, \aleph_P \cup \aleph_Q \cup \aleph_Z) \mid (s \quad X, \aleph_P) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s \quad Y, \aleph_Q) \in \mathcal{A}_{FT}\ Q\ \sigma \wedge \Sigma s \subseteq X \cup Y$$
$$\qquad \wedge\ \Sigma\aleph_P \subseteq X \wedge \Sigma\aleph_Q \subseteq Y \wedge \Sigma\aleph_Z \subseteq \Sigma \setminus X \setminus Y\}$$

$$\mathcal{A}_{FT}\ P \longleftarrow Q\ \sigma, \mathcal{A}_{FT}\ P \longrightarrow Q\ \sigma \subseteq$$
$$\quad \{(s,\aleph) \mid (s_P, \aleph) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s_Q, \aleph) \in \mathcal{A}_{FT}\ Q\ \sigma \wedge s \in s_P \quad s_Q\}$$

$$\mathcal{A}_{FT}\ P \underset{C}{|||} Q\ \sigma, \mathcal{A}_{FT}\ P \underset{C}{|||} Q\ \sigma \subseteq$$
$$\quad \{(s,\aleph) \mid (s_P, \aleph_P) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s_Q, \aleph_Q) \in \mathcal{A}_{FT}\ Q\ \sigma \wedge s \in s_P \underset{C}{\|} s_Q$$
$$\qquad \wedge\ \aleph \quad C = (\aleph_P \cup \aleph_Q) \quad C \wedge \aleph \setminus C = (\aleph_P \cap \aleph_Q) \setminus C\}$$

$$\mathcal{A}_{FT}\ P \setminus X\ \sigma = \{(s \setminus X, \aleph) \mid (s, \aleph \cup [0, \mathit{end}(s,\aleph)) \times X) \in \mathcal{A}_{FT}\ P\ \sigma\}$$

$$\mathcal{A}_{FT}\ f(P)\ \sigma = \{(f(s), \aleph) \mid (s, f^{-1}(\aleph)) \in \mathcal{A}_{FT}\ P\ \sigma\}$$

$$\mathcal{A}_{FT}\ P \overset{t}{\quad} Q\ \sigma \subseteq$$
$$\quad \{(s,\aleph) \mid \mathit{begin}\ s \quad t \wedge (s,\aleph) \in \mathcal{A}_{FT}\ P\ \sigma\}$$
$$\quad \cup\{(s,\aleph) \mid \mathit{begin}\ s \quad t + \delta \wedge (\langle\rangle, \aleph \quad t) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s,\aleph) - t - \delta \in \mathcal{A}_{FT}\ Q\ \sigma\}$$

$$\mathcal{A}_{FT}\ P \underset{t}{\quad} Q\ \sigma \subseteq$$
$$\quad \{(s,\aleph) \mid \mathit{begin}(s \quad t) \quad t + \delta \wedge (s \quad t, \aleph \quad t) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s,\aleph) - t - \delta \in \mathcal{A}_{FT}\ Q\ \sigma\}$$

$$\mathcal{A}_{FT}\ P \underset{e}{\nabla} Q\ \sigma \subseteq$$
$$\quad \{(s,\aleph) \mid e \notin \Sigma(s,\aleph) \wedge (s,\aleph) \in \mathcal{A}_{FT}\ P\ \sigma\}$$
$$\quad \cup\{(s,\aleph) \mid \exists t \quad s \quad t \quad e = \langle(t,e)\rangle \wedge e \notin \Sigma(\aleph \quad t) \wedge \mathit{begin}(s \quad t) \quad t + \delta$$
$$\qquad \wedge (s \quad t \setminus e, \aleph \quad t) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s,\aleph) - t - \delta \in \mathcal{A}_{FT}\ Q\ \sigma\}$$

♡

The reader will have spotted a great similarity between the expressions in this theorem and the semantic equations for the Timed Failures Model; this is because in developing the semantic equations for the Prioritized Model, we have at all times tried to follow the Failures Model. In some places in the above the relationships are those of inclusion rather than equality; this is a result of our operators being refinements of the corresponding **TCSP** operators.

Proof: Most of the proofs are straightforward; we prove three cases for illustration.

Case external choice:

Let $\sigma = \theta_B \circ \rho$. Then we have

$$
\begin{aligned}
& \mathcal{A}_{FT}\ P \,\Box\, Q\ \sigma \\
= & \left\langle \text{definition} \right\rangle \\
& \theta_B(\{(\tau, \sqsubseteq_P \Box \sqsubseteq_Q, \prec\succ) \mid (\tau, \sqsubseteq_P, \prec\succ) \in \mathcal{A}_{BT}\ P\ \rho \wedge (\tau, \sqsubseteq_Q, \prec\succ) \in \mathcal{A}_{BT}\ Q\ \rho\} \\
& \quad \cup \{(\tau, \sqsubseteq_P \Box \sqsubseteq_Q, s) \mid s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge (\tau, \sqsubseteq_P, s) \in \mathcal{A}_{BT}\ P\ \rho \\
& \qquad\quad \wedge (t, \sqsubseteq_Q, \prec\succ) \in \mathcal{A}_{BT}\ Q\ \rho \wedge (s \uparrow t \sqsupset_P (t, \{\!|\,|\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_Q)\} \\
& \quad \cup \{(\tau, \sqsubseteq_P \Box \sqsubseteq_Q, s) \mid s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge (t, \sqsubseteq_P, \prec\succ) \in \mathcal{A}_{BT}\ P\ \rho \\
& \qquad\quad \wedge (\tau, \sqsubseteq_Q, s) \in \mathcal{A}_{BT}\ Q\ \rho \wedge s \uparrow t \not\sqsupset_P (t, \{\!|\,|\!\})\}) \\
\subseteq & \left\langle \text{definition of } \theta_B \right\rangle \\
& \{(\langle\rangle, \aleph) : TF \mid \aleph \subseteq \mathit{ref}\,(\tau, \sqsubseteq_P \Box \sqsubseteq_Q, \prec\succ) \\
& \quad \wedge (\tau, \sqsubseteq_P, \prec\succ) \in \mathcal{A}_{BT}\ P\ \rho \wedge (\tau, \sqsubseteq_Q, \prec\succ) \in \mathcal{A}_{BT}\ Q\ \rho\} \\
& \cup \{(s', \aleph) : TF \mid s' \neq \langle\rangle \wedge \mathit{begin}\, s' = t \wedge (s', \aleph) \simeq (\tau, \sqsubseteq_P \Box \sqsubseteq_Q, s) \\
& \qquad \wedge (\tau, \sqsubseteq_P, s) \in \mathcal{A}_{BT}\ P\ \rho \wedge (t, \sqsubseteq_Q, \prec\succ) \in \mathcal{A}_{BT}\ Q\ \rho\} \\
& \cup \{(s', \aleph) : TF \mid s' \neq \langle\rangle \wedge \mathit{begin}\, s' = t \wedge (s', \aleph) \simeq (\tau, \sqsubseteq_P \Box \sqsubseteq_Q, s) \\
& \qquad \wedge (t, \sqsubseteq_P, \prec\succ) \in \mathcal{A}_{BT}\ P\ \rho \wedge (\tau, \sqsubseteq_Q, s) \in \mathcal{A}_{BT}\ Q\ \rho\} \\
\subseteq & \left\langle \text{definitions of } \mathit{ref},\ \sqsubseteq_P \Box \sqsubseteq_Q \right\rangle \\
& \{(\langle\rangle, \aleph) \mid (\langle\rangle, \aleph) \in \mathcal{A}_{FT}\ P\ \sigma \cap \mathcal{A}_{FT}\ Q\ \sigma\} \\
& \cup \{(s, \aleph) \mid s \neq \langle\rangle \wedge \mathit{begin}\, s = t \wedge (s, \aleph) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (\langle\rangle, \aleph \quad t) \in \mathcal{A}_{FT}\ Q\ \sigma\} \\
& \cup \{(s, \aleph) \mid s \neq \langle\rangle \wedge \mathit{begin}\, s = t \wedge (\langle\rangle, \aleph \quad t) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s, \aleph) \in \mathcal{A}_{FT}\ Q\ \sigma\} \\
= & \left\langle \text{rearranging; part 2 of theorem 6.1.9} \right\rangle \\
& \{(\langle\rangle, \aleph) \mid (\langle\rangle, \aleph) \in \mathcal{A}_{FT}\ P\ \sigma \cap \mathcal{A}_{FT}\ Q\ \sigma\} \\
& \cup \{(s, \aleph) \mid s \neq \langle\rangle \wedge (s, \aleph) \in \mathcal{A}_{FT}\ P\ \sigma \cup \mathcal{A}_{FT}\ Q\ \sigma \\
& \qquad\qquad \wedge (\langle\rangle, \aleph \quad \mathit{begin}\, s) \in \mathcal{A}_{FT}\ P\ \sigma \cap \mathcal{A}_{FT}\ Q\ \sigma\}
\end{aligned}
$$

□

Case hiding:

Firstly, if $\sqsubseteq' \setminus X = \sqsubseteq$, we have

$$
\begin{aligned}
& ref(\tau, \sqsubseteq, s)\\
= \ & \left\langle \text{definition of } ref \right\rangle\\
& \text{closure}\{(t,a) \mid t < \tau \wedge s \uparrow t \uplus (t,a) \not\sqsupset s \uparrow t\}\\
= \ & \left\langle \text{definition of } \sqsubseteq' \setminus X \right\rangle\\
& \text{closure}\{(t,a) \mid t < \tau \wedge (s \uparrow t \uplus (t,a) \notin \text{items} \sqsubseteq \vee \Uparrow_{\sqsubseteq'}^{-\setminus X} (s \uparrow t \uplus (t,a)) \sqsubset' \Uparrow_{\sqsubseteq'}^{-\setminus X} (s \uparrow t))\}\\
\subseteq \ & \left\langle \text{definition of } \Uparrow_{\sqsubseteq'}^{-\setminus X} \right\rangle\\
& \text{closure}\{(t,a) \mid t < \tau \wedge ((\Uparrow_{\sqsubseteq'}^{-\setminus X} (s \uparrow t)) \uplus (t,a) \notin \text{items} \sqsubseteq'\\
& \qquad\qquad\qquad\qquad \vee (\Uparrow_{\sqsubseteq'}^{-\setminus X} (s \uparrow t)) \uplus (t,a) \sqsubset' \Uparrow_{\sqsubseteq'}^{-\setminus X} (s \uparrow t))\}\\
= \ & \left\langle \text{rearranging} \right\rangle\\
& \text{closure}\{(t,a) \mid t < \tau \wedge (\Uparrow_{\sqsubseteq'}^{-\setminus X} s) \uparrow t \uplus (t,a) \not\sqsupset' (\Uparrow_{\sqsubseteq'}^{-\setminus X} s) \uparrow t\}\\
= \ & \left\langle \text{definition} \right\rangle\\
& ref(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus X} s)
\end{aligned}
$$

Now, let $\sigma = \theta_B \circ \rho$. Then

$$
\begin{aligned}
& \mathcal{A}_{FT}\ P \setminus X\ \sigma\\
= \ & \left\langle \text{definition} \right\rangle\\
& \theta_B\{(\tau, \sqsubseteq, s) \mid \forall t \quad s \uparrow t \in \text{items} \sqsubseteq \wedge \exists \sqsubseteq' \quad \sqsubseteq' \setminus X = \sqsubseteq \wedge (\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus X} s) \in \mathcal{A}_{BT}\ P\ \rho\}\\
= \ & \left\langle \text{definition of } \theta_B \right\rangle\\
& \{(s', \aleph) : TF \mid (s', \aleph) \simeq (\tau, \sqsubseteq, s) \wedge \forall t \quad s \uparrow t \in \text{items} \sqsubseteq\\
& \qquad \wedge \exists \sqsubseteq' \quad \sqsubseteq' \setminus X = \sqsubseteq \wedge (\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus X} s) \in \mathcal{A}_{BT}\ P\ \rho\}\\
\subseteq \ & \left\langle \text{using the above result} \right\rangle\\
& \{(s', \aleph) : TF \mid s' \sim s \wedge \exists \sqsubseteq' \quad \aleph \subseteq ref(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus X} s) \wedge (\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus X} s) \in \mathcal{A}_{BT}\ P\ \rho\}\\
\subseteq \ & \left\langle \begin{array}{l} \text{putting } s''' = \Uparrow_{\sqsubseteq'}^{-\setminus X} s,\ s' = s'' \setminus X \text{ for appropriate } s'' \text{ such that } s'' \sim s''';\\ [0, end(s, \aleph)) \times X \subseteq ref(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus X} s) \text{ by definition of } \Uparrow_{\sqsubseteq'}^{-\setminus X} s \end{array} \right\rangle\\
& \{(s'' \setminus X, \aleph) \mid s'' \sim s''' \wedge \aleph \in RSET\\
& \qquad \wedge \exists \sqsubseteq' \quad \aleph \cup [0, end(s, \aleph)) \times X \subseteq ref(\tau, \sqsubseteq', s''') \wedge (\tau, \sqsubseteq', s''') \in \mathcal{A}_{BT}\ P\ \rho\}\\
= \ & \left\langle \text{definition of } \mathcal{A}_{FT} \right\rangle\\
& \{(s'' \setminus X, \aleph) \mid (s'', \aleph \cup [0, end(s, \aleph)) \times X) \in \mathcal{A}_{FT}\ P\ \sigma\}
\end{aligned}
$$

□

Case variables:

Let $\sigma = \theta_B \circ \rho$; then $\mathcal{A}_{FT}\ X\ \sigma = \theta_B(\rho\ X\ ) = \sigma\ X$. □

This completes the proof. □

6.1.4 The abstraction result

We are now able to prove our abstraction result.

Theorem 6.1.14: $\forall P : BTCSP \quad \mathcal{A}_{FT}\ P\ \sigma \subseteq \mathcal{F}_T\ \varphi P\ \sigma$ ♡

Proof: We prove the result by structural iuduction. All cases except for recursion follow easily from theorem 6.1.13. We give the proofs for parallel composition as an example.

Case parallel composition:

$$
\begin{array}{ll}
 & (s,\aleph) \in \mathcal{A}_{FT}\ P \mathbin{\#\!\!\!\!\#} Q\ \sigma) \\
\Rightarrow & \left\langle \text{theorem 6.1.13} \right\rangle \\
 & \exists \aleph_P, \aleph_Q \quad \aleph = \aleph_P \cup \aleph_Q \wedge (s,\aleph_P) \in \mathcal{A}_{FT}\ P\ \sigma \wedge (s,\aleph_Q) \in \mathcal{A}_{FT}\ Q\ \sigma \\
\Rightarrow & \left\langle \text{inductive hypothesis} \right\rangle \\
 & \exists \aleph_P, \aleph_Q \quad \aleph = \aleph_P \cup \aleph_Q \wedge (s,\aleph_P) \in \mathcal{F}_T\ \varphi P\ \sigma \wedge (s,\aleph_Q) \in \mathcal{F}_T\ \varphi Q\ \sigma \\
\Leftrightarrow & \left\langle \text{definition of parallel composition in } \mathcal{M}_{TF} \right\rangle \\
 & (s,\aleph) \in \mathcal{F}_T\ \varphi P \parallel \varphi Q\ \sigma \\
\Leftrightarrow & \left\langle \text{definition of } \varphi \right\rangle \\
 & (s,\aleph) \in \mathcal{F}_T\ \varphi(P \mathbin{\#\!\!\!\!\#} Q)\ \sigma
\end{array}
$$

□

We now prove the result for recursion.

Case immediate recursion:

Let $\sigma = \theta_B \circ \rho$. Then we have

$$
\begin{array}{ll}
 & \mathcal{A}_{FT}\ \mu X \quad P\ \sigma \\
= & \left\langle \text{definition} \right\rangle \\
 & \theta_B(\mathit{fix}(M_A(X,P)\rho)) \\
= & \left\langle \text{Banach's fixed point theorem} \right\rangle \\
 & \theta_B\left(\lim_{n\to\infty}(M_A(X,P)\rho)^n(STOP_B)\right) \\
= & \left\langle \text{continuity of } \theta_B \right\rangle \\
 & \lim_{n\to\infty} \theta_B((M_A(X,P)\rho)^n(STOP_B))
\end{array}
$$

where $M_A(X,P)\rho = \lambda Y \quad \mathcal{A}_{BT}\ P\ \rho[Y/X]$ and $STOP_B = \mathcal{A}_{BT}\ STOP\ \rho$. Similarly,

$$\mathcal{F}_T\ \varphi(\mu X \quad P)\ \sigma = \lim_{n\to\infty}(M_F(X,\varphi P)\sigma)^n(STOP_F)$$

where $M_F(X,Q)\sigma = \lambda Y \quad \mathcal{F}_T\ Q\ \sigma[Y/X]$ and $STOP_F = \mathcal{F}_T\ STOP\ \sigma$. To prove our result, we make use of the following lemma:

Lemma 6.1.14.1: If $\sigma = \theta_B \circ \rho$, then for all natural numbers n

$$\theta_B((M_A(X,P)\rho)^n(STOP_B)) \subseteq (M_F(X,\varphi P)\sigma)^n(STOP_F)$$

♡

Proof of lemma: We proceed by numerical induction. The base case follows immediately from theorem 6.1.13. For the inductive step, assume that

$$\theta_B((M_A(X,P)\rho)^n(STOP_B)) \subseteq (M_F(X,\varphi P)\sigma)^n(STOP_F)$$

Then we have

$$\begin{array}{ll}
& \theta_B((M_A(X,P)\rho)^{n+1}(STOP_B)) \\
= & \left\langle \text{rearranging} \right\rangle \\
& \theta_B(M_A(X,P)\rho((M_A(X,P)\rho)^n(STOP_B))) \\
= & \left\langle \text{definition of } M_A(X,P)\rho \right\rangle \\
& \theta_B(\mathcal{A}_{BT}\ P\ \rho[(M_A(X,P)\rho)^n(STOP_B)/X]) \\
= & \left\langle \text{definition of } \mathcal{A}_{FT} \right\rangle \\
& \mathcal{A}_{FT}\ P\ \sigma[\theta_B((M_A(X,P)\rho)^n(STOP_B))/X] \\
\subseteq & \left\langle \text{structural inductive hypothesis} \right\rangle \\
& \mathcal{F}_T\ \varphi P\ \sigma[\theta_B((M_A(X,P)\rho)^n(STOP_B))/X] \\
\subseteq & \left\langle \begin{array}{l} \text{numerical inductive hypothesis;} \\ \text{monotonicity of } \mathcal{F}_T\ \varphi P \text{ with respect to the subset relation} \end{array} \right\rangle \\
& \mathcal{F}_T\ \varphi P\ \sigma[(M_F(X,\varphi P)\sigma)^n(STOP_F)/X] \\
= & \left\langle \text{definition of } M_F(X,\varphi P)\sigma \right\rangle \\
& M_F(X,\varphi P)\sigma((M_F(X,\varphi P)\sigma)^n(STOP_F)) \\
= & \left\langle \text{rearranging} \right\rangle \\
& (M_F(X,\varphi P)\sigma)^{n+1}(STOP_F)
\end{array}$$

□

Hence, by continuity of the subset relation, we have

$$\lim_{n\to\infty} \theta_B((M_A(X,P)\rho)^n(STOP_B)) \subseteq \lim_{n\to\infty} (M_F(X,\varphi P)\sigma)^n(STOP_F)$$

So we have shown

$$\mathcal{A}_{FT}\ \mu X \quad P\ \sigma \subseteq \mathcal{F}_T\ \varphi(\mu X \quad P)\ \sigma$$

□

Case mutual recursion:

We only consider the case where the vector of terms $\underline{P}$ is constructive for the vector of terms $\underline{X}$. Recall from chapter 4 that

$$\mathcal{A}_{BT}\ \langle X_i = P_i \rangle_j\ \rho \mathrel{\hat{=}} S_j \text{ where } \underline{S} \text{ is a unique fixed point of } M(\underline{X},\underline{P})\rho$$

In that chapter we defined a subsidiary vector $\underline{Q}$ by

$$Q_i \mathrel{\hat=} P_i[Q_j/X_j \mid j \in seq(i)]$$

and showed that $M(\underline{X}, \underline{Q})\rho$ is a contraction mapping whose unique fixed point is also the unique fixed point of $M(\underline{X}, \underline{P})\rho$. As in the previous case, we can show that

$$\mathcal{A}_{FT} \; \langle X_i = P_i \rangle_j \; \sigma = \lim_{n\to\infty} \theta_B \left((M_A(\underline{X}, \underline{Q})\rho)^n (\underline{STOP_B}) \right)_j$$

where $\underline{STOP_B} \mathrel{\hat=} \langle \mathcal{A}_{BT} \; STOP \; \rho \mid i \in I \rangle$; and

$$\mathcal{F}_T \; \varphi(\langle X_i = P_i \rangle_j) \; \sigma = \lim_{n\to\infty} \left((M_F(\underline{X}, \varphi\underline{Q})\sigma)^n (\underline{STOP_F}) \right)_j$$

where $\underline{STOP_F} = \langle \mathcal{F}_T \; STOP \; \sigma \mid i \in I \rangle$. As in the previous case, it is easy to show

$$\theta_B \left((M_A(\underline{X}, \underline{Q})\rho)^n (\underline{STOP_B}) \right)_j \subseteq \left((M_F(\underline{X}, \varphi\underline{Q})\sigma)^n (\underline{STOP_F}) \right)_j$$

(for all $j \in I$) by numerical induction, thus completing the case. □

This completes the proof. □

6.1.5 On recursion

In this section we study the semantic value of the recursive process $\mu X \;\; P$ in the space $\mathcal{M}_{TF}$. We will show that if $\sigma = \theta_B \circ \rho$ then $\mathcal{A}_{FT} \; \mu X \;\; P \; \sigma$ is the unique fixed point of the relation $\theta_B \circ M_A(X, P)\rho \circ \theta_B^{-1}$.

Note that $\theta_B \circ M_A(X, P)\sigma \circ \theta_B^{-1}$ is not always a function. Let $X_1 \mathrel{\hat=} \mathcal{A}_{BT} \; (b \sqcap d \sqcap a) \setminus d \; \rho$ and $X_2 \mathrel{\hat=} \mathcal{A}_{BT} \; (a \sqcap d \sqcap b) \setminus d \; \rho$. Then $\theta_B(X_1) = \theta_B(X_2)$; call this image '$Y$'. However, consider $M_A(X, P)\rho(X_1)$ and $M_A(X, P)\rho(X_2)$ where $P \mathrel{\hat=} X \sqcap (b \longleftarrow c)$. Note that these are both members of $(M_A(X, P)\rho \circ \theta_B^{-1})(Y)$. It is easy to see that $M_A(X, P)\rho(X_1)$ can perform a b and refuse a c (having an offer relation with $\{\!|b|\!\} \sqsupset \{\!|b, c|\!\} \sqsupset \{\!|c|\!\} \sqsupset \{\!|\,|\!\} \sqsupset \{\!|a|\!\}$ initially), whereas $M_A(X, P)\rho(X_2)$ cannot (having an offer relation with $\{\!|a|\!\} \sqsupset \{\!|b, c|\!\} \sqsupset \{\!|b|\!\} \sqsupset \{\!|c|\!\} \sqsupset \{\!|\,|\!\}$ initially, and $\{\!|c|\!\} \sqsupset \{\!|\,|\!\}$ after performing a b). Hence we have $\theta_B(M_A(X, P)\rho(X_1)) \neq \theta_B(M_A(X, P)\rho(X_2))$. So $(\theta_B \circ M_A(X, P)\rho)(X_1)$ and $(\theta_B \circ M_A(X, P)\rho)(X_2)$ are distinct members of $(\theta_B \circ M_A(X, P)\rho \circ \theta_B^{-1})(Y)$.

We show that $\mathcal{A}_{FT} \; \mu X \;\; P \; \sigma$ is a fixed point of $\theta_B \circ M_A(X, P)\rho \circ \theta_B^{-1}$:

Lemma 6.1.15: If $\sigma = \theta_B \circ \rho$ then

$$\mathcal{A}_{FT} \; \mu X \;\; P \; \sigma \in (\theta_B \circ M_A(X, P)\rho \circ \theta_B^{-1})(\mathcal{A}_{FT} \; \mu X \;\; P \; \sigma)$$

♡

Proof: Let $Q \mathrel{\hat=} \mathcal{A}_{FT} \; \mu X \;\; P \; \sigma$ and let $Q' \mathrel{\hat=} \mathcal{A}_{BT} \; \mu X \;\; P \; \rho$. Then from the definitions of θ_B and recursion we have $Q' \in \theta_B^{-1}(Q)$ and $Q \in (\theta_B \circ M_A(X, P)\rho)(Q')$ so we have $Q \in (\theta_B \circ M_A(X, P)\sigma \circ \theta_B^{-1})(Q)$. □

To show that $\mathcal{A}_{FT}\ \mu X \quad P\ \sigma$ is the *unique* fixed point is a little harder. Recall that we only define the recursive term $\mu X \quad P$ for terms P that are constructive for the variable X, where P is t-constructive for X in $\mathcal{M}_{TB}$ if

$$\forall t_0 : TIME\ ; \rho : ENV_B \quad \mathcal{A}_{BT}\ P\ \rho \quad t_0 + t = \mathcal{A}_{BT}\ P\ \rho[\rho\ X \quad t_0/X] \quad t_0 + t$$

We shall say that BTCSP term P is t-constructive for X in $\mathcal{M}_{TF}$ if

$$\forall t_0 : TIME\ ; \sigma : ENV_F \quad \mathcal{A}_{FT}\ P\ \sigma \quad t_0 + t = \mathcal{A}_{FT}\ P\ \sigma[\sigma\ X \quad t_0/X] \quad t_0 + t$$

The following lemma relates these two concepts:

Lemma 6.1.16: If P is t-constructive for X in $\mathcal{M}_{TB}$, then P is t-constructive for X in $\mathcal{M}_{TF}$. ♡

Proof: Note that for any $Y \in \mathcal{M}_{TB}$ we have

$$\theta_B(Y) \quad t = \theta_B(Y \quad t) \qquad (*)$$

by the definition of θ_B. Suppose then that P is t-constructive for X in $\mathcal{M}_{TB}$ and let $\sigma = \theta_B \circ \rho$; we have

$$\begin{aligned}
& \mathcal{A}_{FT}\ P\ \sigma[\sigma\ X \quad t_0/X] \quad t_0 + t \\
= & \left\langle \text{definition of } \mathcal{A}_{FT}\text{, using } (*) \text{ applied to } \rho\ X \right\rangle \\
& \theta_B(\mathcal{A}_{BT}\ P\ \rho[\rho\ X \quad t_0/X]) \quad t_0 + t \\
= & \left\langle \text{by } (*) \right\rangle \\
& \theta_B(\mathcal{A}_{BT}\ P\ \rho[\rho\ X \quad t_0/X] \quad t_0 + t) \\
= & \left\langle P \text{ is } t\text{-constructive for } X \text{ in } \mathcal{M}_{TB} \right\rangle \\
& \theta_B(\mathcal{A}_{BT}\ P\ \rho \quad t_0 + t) \\
= & \left\langle \text{by } (*) \right\rangle \\
& \theta_B(\mathcal{A}_{BT}\ P\ \rho) \quad t_0 + t \\
= & \left\langle \text{definition of } \mathcal{A}_{FT} \right\rangle \\
& \mathcal{A}_{FT}\ P\ \sigma \quad t_0 + t
\end{aligned}$$

□

Suppose then that Y is *any* fixed point of $\theta_B \circ M_A(X, P)\rho \circ \theta_B^{-1}$. The following lemma shows that it is also a fixed point of $\lambda Y \quad \mathcal{A}_{FT}\ P\ \sigma[Y/X]$, where $\sigma = \theta_B \circ \rho$.

Lemma 6.1.17: If $\sigma = \theta_B \circ \rho$ and Y is a fixed point of $\theta_B \circ M_A(X, P)\rho \circ \theta_B^{-1}$ then

$$Y = \mathcal{A}_{FT}\ P\ \sigma[Y/X]$$

♡

Proof: For some $Y' \in \mathcal{M}_{TB}$ we have $Y = \theta_B(Y') = (\theta_B \circ M_A(X,P)\rho)(Y')$. Hence we have

$$
\begin{aligned}
& Y \\
= \; & \langle \text{hypothesis} \rangle \\
& (\theta_B \circ M_A(X,P)\rho)(Y') \\
= \; & \langle \text{definition of } M_A(X,P)\rho \rangle \\
& \theta_B(\mathcal{A}_{BT} \;\; P \;\; \rho[Y'/X]) \\
= \; & \langle \text{definition of } \mathcal{A}_{FT};\ Y = \theta_B(Y') \rangle \\
& \mathcal{A}_{FT} \;\; P \;\; \sigma[Y/X]
\end{aligned}
$$

□

We can now show that the fixed point is unique.

Theorem 6.1.18: $\mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma$ is the unique fixed point of $\theta_B \circ M_A(X,P)\rho \circ \theta_B^{-1}$ where $\sigma = \theta_B \circ \rho$. ♡

Proof: We have already shown that $\mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma$ is a fixed point of $\theta_B \circ M_A(X,P)\rho \circ \theta_B^{-1}$. For uniqueness, suppose $Y \in \mathcal{M}_{TF}$ is an arbitrary fixed point. Suppose that P is t-constructive for X and assume that

$$Y \quad t_0 = \mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma \quad t_0 \tag{$*$}$$

it is enough to show that $Y \quad t_0 + t = \mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma \quad t_0 + t$. We have

$$
\begin{aligned}
& Y \quad t_0 + t \\
= \; & \langle \text{previous lemma} \rangle \\
& \mathcal{A}_{FT} \;\; P \;\; \sigma[Y/X] \quad t_0 + t \\
= \; & \langle P \text{ is } t\text{-constructive for } X \rangle \\
& \mathcal{A}_{FT} \;\; P \;\; \sigma[Y \quad t_0/X] \quad t_0 + t \\
= \; & \langle \text{from } (*) \rangle \\
& \mathcal{A}_{FT} \;\; P \;\; \sigma[\mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma \quad t_0/X] \quad t_0 + t \\
= \; & \langle P \text{ is } t\text{-constructive for } X \rangle \\
& \mathcal{A}_{FT} \;\; P \;\; \sigma[\mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma/X] \quad t_0 + t \\
= \; & \langle \text{previous lemma applied to } \mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma \rangle \\
& \mathcal{A}_{FT} \;\; \mu X \quad P \;\; \sigma \quad t_0 + t
\end{aligned}
$$

as required. □

6.2 Using the abstraction result to simplify proofs

We now prove a result which will allow us to translate specifications on BTCSP processes into specifications on TCSP processes. We claim that the failures specification $S(s,\aleph)$ can be translated into the priorities specification $\Theta_B S(\tau,\sqsubseteq,s)$, where we define the mapping $\Theta_B : (TF \to Bool) \to (BEH \to Bool)$ by:

Definition 6.2.1: $\Theta_B S(\tau,\sqsubseteq,s) \mathrel{\widehat{=}} \forall (s',\aleph) : TF \quad (s',\aleph) \simeq (\tau,\sqsubseteq,s) \Rightarrow S(s',\aleph)$. ◇

The specification $\Theta_B S$ is true of a behaviour $(\tau, \sqsubseteq, s)$ if all corresponding failures $(s', \aleph)$ satisfy $S(s', \aleph)$.

We can now state our abstraction resnlt.

Rnle 6.2.2 (Abstraction)

$$\frac{\varphi_B(P)\ \mathbf{sat}_\sigma\ S(s,\aleph)\ \mathbf{in}\ \mathcal{M}_{TF}}{P\ \mathbf{sat}_\rho\ \Theta_B S(\tau,\sqsubseteq,s)\ \mathbf{in}\ \mathcal{M}_{TB}}\Big[\sigma = \theta_B \circ \rho\Big]$$

$\triangle$

If a TCSP process satisfies specification $S(s, \aleph)$, then all its prioritized refinements satisfy the specification $\Theta_B S(\tau, \sqsubseteq, s)$. Put another way, in order to show that a BTCSP process P satisfies a specification $S'(\tau, \sqsubseteq, s)$, we need to find a failures specifieation $S(s, \aleph)$ such that $\Theta_B S = S'$, and then use the proof rules for the Failures Model to show that the TCSP abstraction of P satisfies $S(s, \aleph)$.

Proof: Assume the premise. Then we have hy the definition of **sat**:

$$\begin{array}{ll}
 & \forall (s',\aleph) : TF \quad (s',\aleph) \in \mathcal{F}_T\ \varphi_B(P)\ \sigma \Rightarrow S(s',\aleph) \\
\Rightarrow & \Big\langle \text{theorem 6.1.14 using the side condition} \Big\rangle \\
 & \forall (s',\aleph) : TF \quad (s',\aleph) \in \theta_B(\mathcal{A}_{BT}\ P\ \rho) \Rightarrow S(s',\aleph) \\
\Leftrightarrow & \Big\langle \text{definition of } \theta_B \Big\rangle \\
 & \forall (s',\aleph) : TF \quad (\exists (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ P\ \rho \quad (s',\aleph) \simeq (\tau,\sqsubseteq,s)) \Rightarrow S(s',\aleph) \\
\Leftrightarrow & \Big\langle \text{predicate calculus} \Big\rangle \\
 & \forall (\tau,\sqsubseteq,s) \in \mathcal{A}_{BT}\ P\ \rho \quad \forall (s',\aleph) : TF \quad (s',\aleph) \simeq (\tau,\sqsubseteq,s) \Rightarrow S(s',\aleph) \\
\Leftrightarrow & \Big\langle \text{definition of } \mathbf{sat}\text{; definition of } \Theta_B S \Big\rangle \\
 & P\ \mathbf{sat}_\rho\ \Theta_B S(\tau,\sqsubseteq,s)
\end{array}$$

$\square$

The following version of the rule will prove to he more useful:

Rule 6.2.3:

$$\frac{\begin{array}{l}\varphi_B(P)\ \mathbf{sat}_\sigma\ S(s,\aleph)\ \mathbf{in}\ \mathcal{M}_{TF} \\ \Theta_B S(\tau,\sqsubseteq,s) \Rightarrow S'(\tau,\sqsubseteq,s)\end{array}}{P\ \mathbf{sat}_\rho\ S'(\tau,\sqsubseteq,s)\ \mathbf{in}\ \mathcal{M}_{TB}}\Big[\sigma = \theta_B \circ \rho\Big]$$

$\triangle$

This can be proved using the previous rule and rule B.1.3.

The following rule provides a way of redueing proof obligations on probabilistic processes to proof obligations on processes in the Failures Model.

Rule 6.2.4:

$$\frac{\begin{array}{l}(\varphi_B \circ \varphi_P^{(B)})(P)\ \mathbf{sat}_\sigma\ S(s,\aleph)\ \mathbf{in}\ \mathcal{M}_{TF} \\ \Theta_B S(\tau,\sqsubseteq,s) \Rightarrow S'(\tau,\sqsubseteq,s)\end{array}}{P\ \mathbf{sat}_\rho\ S'(\tau,\sqsubseteq,s)\ \mathbf{in}\ \mathcal{M}_{PTB}}\Big[\sigma = \theta_B \circ \pi_I \circ \rho\Big]$$

$\triangle$

This can be proved using the previous rule and the abstraction rule from section 5.2. Note that $\varphi_B \circ \varphi_P^{(B)}$ is the mapping that removes all probabilities *and* priorities from the syntax of PBTCSP.

To make it easier to use these rules, we would like ways of translating specifications from the Prioritized Model to the Failures Model: given a specification $S'(\tau, \sqsubseteq, s)$ we want to be able to find a corresponding specification $S(s, \aleph)$ such that $\Theta_B S \Rightarrow S'$. In the next section we develop a number of rules for aiding us in this.

6.2.1 Translation of priorities specifications into failures specifications

In this subsection we investigate which specifications translate easily under Θ_B: given a specification $S'(\tau, \sqsubseteq, s)$ we want to be able to find a specification $S(s, \aleph)$ such that $\Theta_B S(\tau, \sqsubseteq, s) \Rightarrow S'(\tau, \sqsubseteq, s)$. In particular, we give a number of results which show that many predicates written in our specification language will not change form when transformed by Θ_B; for example, we will show that $\Theta_B(a \text{ at } t \Rightarrow b \text{ live from } t+1) \Rightarrow (a \text{ at } t \Rightarrow b \text{ live from } t+1)$. Most of the results of this section were proved in [Low92b].

The at operator is preserved by Θ_B since if $s \sim s'$ then s and s' contain the same events.

Lemma 6.2.5: $\Theta_B(A \text{ at}^n\ I) = A \text{ at}^n\ I$ and $\Theta_B(\text{no } A \text{ at}^n\ I) = \text{no } A \text{ at}^n\ I$. ♡

Our result for the live operator is slightly weaker:

Lemma 6.2.6: $\Theta_B(A \text{ live}^n\ I) \Rightarrow A \text{ live}^n\ I$. ♡

Fortunately this implication is strong enough for our purposes so long as we do not use live in negated form or on the left hand side of implications. If the interval I is open on the right then we have a stronger result:

Lemma 6.2.7: If I is open on the right then $\Theta_B(A \text{ live}^n\ I) = A \text{ live}^n\ I$. ♡

So in particular

Lemma 6.2.8: $\Theta_B(A \text{ live}^n \text{ from } t) = A \text{ live}^n \text{ from } t$. ♡

For the beyond macro, note that $(s', \aleph) \simeq (\tau, \sqsubseteq, s) \Rightarrow end(s', \aleph) \quad \tau$ so we have

Lemma 6.2.9: $\Theta_B(\text{beyond } t) \Rightarrow \text{beyond } t$. ♡

History predicates

Many of our specifications are of the form $S \mathrel{\hat{=}} \varphi \circ M$ where M is a projection mapping from traces in the prioritized model to some type T, and φ is a predicate on T. If there is a similar projection mapping function M' from traces in the Timed Failures Model to T, giving the same value as M on related traces, then S translates to $\varphi \circ M'$.

Lemma 6.2.10: If the projection mappings $M : TT \to T$ and $M' : T\Sigma^*_{\leqslant} \to T$ are such that

$$s' \sim s \Rightarrow M(s) = M'(s')$$

then $\varphi \circ M = \Theta_B(\varphi \circ M')$. ♡

We will be particnlarly interested in those mappings M and M' that take the same form in our two specification languages. For example, the projection functions count and alphabet do, so we have for example

$$\begin{aligned}\Theta_B(\mathsf{count}\, A \text{ during } I < 3) &= \mathsf{count}\, A \text{ during } I < 3\\ \Theta_B(\mathsf{alphabet} \subseteq A) &= \mathsf{alphabet} \subseteq A\end{aligned}$$

The operators first and last need some care. In the Prioritized Model these operators return a pair consisting of a time and an action (which could contain more than one event), whereas in the Failures Model they return timed events, i.e. pairs consisting of a time and a *single* event. However, we have

$$s' \sim s \Rightarrow (\mathsf{first}\, A \text{ during } I)(s') \in (\mathsf{first}\, A \text{ during } I)(s)$$

and so

$$\Theta_B(\mathsf{first}\, A \text{ during } I = (t, a)) \;=\; (t, a) \in \mathsf{first}\, A \text{ during } I$$

where we define the $\in$ operator on offers by $(t, a) \in (t', \alpha) \Leftrightarrow t = t' \wedge a \in \alpha$. The following lemma is slightly stronger. Let a^n denote the action containing n as.

Lemma 6.2.11: $\Theta_B(\mathsf{first}\, A \text{ during } I = (t, a)) = \exists\, n : {}^{+} \;\; \mathsf{first}\, A \text{ during } I = (t, a^n)$. ♡

If the first timed event of a trace in the Failures Model is (t, a), then the trace of the corresponding prioritized behaviour must have started with a number of as at time t. Note that some other part of the specification will often be enough to ensure that only one a occurs. A similar result holds for the last macro.

The name of and time of operators behave as one would expect. We have

$$\begin{aligned}\Theta_B(\text{time of first}\, A \text{ during } I = t) &= \text{time of first}\, A \text{ during } I = t\\ \Theta_B(\text{name of first}\, A \text{ during } I = a) &= \exists\, n : {}^{+} \;\; \text{name of first}\, A \text{ during } I = a^n\end{aligned}$$

Similar results hold for the last operator or when the '=' is replaced by an inequality; so, for example, we have

$$\Theta_B(\text{time of last}\, A \text{ during } I \quad 3) = \text{time of last}\, A \text{ during } I \quad 3$$

Environmental assumptions

Recall the definition of the environmental condition internal in the Failures Model:

$$(\mathsf{internal}\, A)(s, \aleph) \mathrel{\widehat{=}} [0, end(s, \aleph)) \times A \subseteq \aleph$$

If we calculate $\Theta_B(\mathsf{internal}\, A)(\tau, \sqsubseteq, s)$ we see that it is equivalent to *false* (for $A \neq \{\}$) because $\Theta_B(\mathsf{internal}\, A)(\tau, \sqsubseteq, s)$ is the condition that *all* refusal sets relating to behaviour $(\tau, \sqsubseteq, s)$ — including the empty refusal — contain the elements of A at all times. A similar result holds for the open and accessible operators. Thus we find that we cannot translate these predicates directly.

However environmental conditions are normally used on the left hand side of implications, for example in specifications such as $\mathsf{internal}\, A \Rightarrow a$ at 2. It is normally the case that the consequent of the implication does not talk about the elements of A being refused. In this case we can show that $\Theta_B(\mathsf{internal}\, A \Rightarrow S)$ implies $\mathsf{internal}\, A \Rightarrow \Theta_B S$. We say that the specification $S(s, \aleph)$ is A-refusal independent if the addition of elements of A to the refusal set makes no difference to the truth of S.

Definition 6.2.12: The specification $S : TF \rightarrow \mathit{Bool}$ is A-refusal independent iff

$$\forall (s, \aleph) : TF \quad \forall \aleph' : RSET \quad \Sigma\aleph' \subseteq A \Rightarrow (S(s, \aleph) \Leftrightarrow S(s, \aleph \cup \aleph'))$$

$\diamondsuit$

In this case, the specification $(\mathsf{internal}\, A \Rightarrow S)(\tau, \sqsubseteq, s)$ translates easily.

Lemma 6.2.13: If S is A-refusal independent then

$$\Theta_B(\mathsf{internal}\, A \Rightarrow S) \Rightarrow (\mathsf{internal}\, A \Rightarrow \Theta_B S)$$

$\heartsuit$

A similar result holds for the open operator. We say that the specification $S(s, \aleph)$ is (A, I)-refusal independent if the addition of elements of A during the interval I to the refusal set makes no difference to the truth of S.

Definition 6.2.14: The specification $S : TF \rightarrow \mathit{Bool}$ is (A, I)-refusal indepeudent iff there is some $J \supseteq I$ such that J is a finite union of half-open time intervals and

$$\forall (s, \aleph) : TF \quad \forall \aleph' : RSET \quad \aleph' \subseteq J \times A \Rightarrow (S(s, \aleph) \Leftrightarrow S(s, \aleph \cup \aleph'))$$

$\diamondsuit$

We then have the following result:

Lemma 6.2.15: If S is (A, I)-refusal independent then

$$\Theta_B(A\ \mathsf{open}^n\ I \Rightarrow S) \Rightarrow (A\ \mathsf{open}^n\ I \Rightarrow \Theta_B S)$$

$\heartsuit$

The closed macro translates very easily:

Lemma 6.2.16: $\Theta_B(A\ \mathsf{closed}\ I) = (A\ \mathsf{closed}\ I)$. $\heartsuit$

The accessible α predicate is highly dependent upon priorities, and so it is harder to translate it into a specification without priorities. We give a partial result for when α is a singleton action. Firstly we define a failures specification accessible by

$$(a\ \mathsf{accessible}\ I)(s, \aleph) \mathrel{\widehat{=}} (\forall t \in I \quad a\ \mathsf{at}\ I \cap [0, t] \vee a\ \mathsf{ref}\ t)(s, \aleph)$$

We have the following result:

Lemma 6.2.17: If the interval I is open on the right, and S is $(\{a\}, I)$-refusal independent then

$$\Theta_B(a\ \mathsf{accessible}\ I \Rightarrow S) \Rightarrow (a\ \mathsf{accessible}\ I \Rightarrow \Theta_B S)$$

$\heartsuit$

In particular we have

Lemma 6.2.18: If S is $(\{a\},[t,\infty))$-refusal independent then

$$\Theta_B(a \text{ accessible from } t \Rightarrow S) \Rightarrow (a \text{ accessible from } t \Rightarrow \Theta_B S)$$

♡

Boolean operators

Recall that we have lifted the booleau operators, so that $(S \wedge S')(s,\aleph) = S(s,\aleph) \wedge S'(s,\aleph)$, for example. The predicate $\Theta_B(S \wedge S')$ is the same as $\Theta_B S \wedge \Theta_B S'$.

Lemma 6.2.19: $\Theta_B(S \wedge S') = \Theta_B S \wedge \Theta_B S'$. ♡

For implication, our result is not quite so strong.

Lemma 6.2.20: $\Theta_B(S \Rightarrow S')(\tau,\sqsubseteq,s) \Rightarrow (\Theta_B S \Rightarrow \Theta_B S')(\tau,\sqsubseteq,s)$. ♡

Luckily this implication is strong enough for use with rule 6.2.3.
For negation, we have a rule of similar strength

Lemma 6.2.21: $\Theta_B(\neg\, S) \Rightarrow \neg\,(\Theta_B S)$ ♡

Unfortunately, we do not have such a result for disjunctions. For example, let

$$S(s,\aleph) \mathrel{\widehat{=}} (0,a) \in \aleph \qquad S'(s,\aleph) \mathrel{\widehat{=}} (0,a) \notin \aleph$$

It is easy to see that $(S \vee S')(s,\aleph) = \mathit{true}$ so $\Theta_B(S \vee S')(\tau,\sqsubseteq,s) = \mathit{true}$. However,

$$\Theta_B S(\tau,\sqsubseteq,s) \;\Leftrightarrow\; \forall \aleph \subseteq \mathit{ref}(\tau,\sqsubseteq,s) \;\; (0,a) \in \aleph \;\Leftrightarrow\; \mathit{false}$$

and

$$\Theta_B S'(\tau,\sqsubseteq,s) \;\Leftrightarrow\; \forall \aleph \subseteq \mathit{ref}(\tau,\sqsubseteq,s) \;\; (0,a) \notin \aleph \;\Leftrightarrow\; (0,a) \notin \mathit{ref}(\tau,\sqsubseteq,s)$$

so $\Theta_B(S \vee S') \not\Rightarrow \Theta_B S \vee \Theta_B S'$.

Summary

These rules will be enough to translate most of our specifications into failure specifications. We are not claiming that this is a complete set of rules for translating specifications — indeed we believe that there are many more such rules. A library of more rules could be built up by pursuing further case studies. Also, whenever we add a new construct to our specification language we will have to give it a definition in both the Prioritized and Failures Models, and investigate how the construct translates from one model to the other.

6.3 An example using the abstraction result

In this section we deal with an example of a clock that will offer a *tick* every second, except for every T seconds when it will prefer a *tock* (where $T > 1$). Although this example may seem rather artificial, we believe that it demonstrates one particular aspect of priorities quite well, namely interrupts: the *tock*s can be seen as interrupting the "normal" behaviour represented by the *tick*s.

When modelling an interrupt mechanism, the interrupting event should be given a higher priority than the thing being interrupted. We need a prioritized model in order to describe this; however, the process being interrupted and the interrupt handler will often not make use of priorities, and so in order to argue about them it is simplest if we use the Timed Failures Model.

The clock can only perform the events *tick* and *tock*:

$$\mathsf{alphabet} \subseteq \{tick, tock\}$$

Initially it will offer both *tick* and *tock*:

$$tick, tock \ \mathsf{live\ from}\ 0$$

It cannot perform two events within one second of each other:

$$tick, tock \ \mathsf{at}\ t \Rightarrow \mathsf{no}\ tick, tock\ \mathsf{at}\ (t, t+1)$$

*tock*s must occur at least T seconds apart:

$$tock\ \mathsf{at}\ t \Rightarrow \mathsf{no}\ tock\ \mathsf{at}\ (t, t+T)$$

If the clock hasn't performed either a *tick* or a *tock* in the last second, then it should offer a *tick* — i.e. it is willing to perform a *tick* one second after the previous event:

$$\mathsf{no}\ tick, tock\ \mathsf{at}\ (t-1, t) \Rightarrow tick\ \mathsf{live}\ t$$

If the clock hasn't performed a *tock* in the last T seconds, and hasn't performed a *tick* in the last second, then it will be willing to perform a *tock*:

$$\mathsf{no}\ tock\ \mathsf{at}\ (t-T, t) \wedge \mathsf{no}\ tick\ \mathsf{at}\ (t-1, t) \Rightarrow tock\ \mathsf{live}\ t$$

If the process is able to perform either a *tick* or a *tock*, then it prefers the *tock* to the *tick*:

$$tick\ \mathsf{offered}\ t \wedge tock\ \mathsf{offered}\ t \Rightarrow tock\ \mathsf{preferred\ to}\ tick\ @\ t$$

Putting these together, we get the following specification:

$$\begin{aligned}
S \mathrel{\widehat=} {} & \mathsf{alphabet} \subseteq \{tick, tock\} \\
& \wedge\ tick, tock\ \mathsf{live\ from}\ 0 \\
& \wedge\ tick, tock\ \mathsf{at}\ t \Rightarrow \mathsf{no}\ tick, tock\ \mathsf{at}\ (t, t+1) \\
& \wedge\ tock\ \mathsf{at}\ t \Rightarrow \mathsf{no}\ tock\ \mathsf{at}\ (t, t+T) \\
& \wedge\ \mathsf{no}\ tick, tock\ \mathsf{at}\ (t-1, t) \Rightarrow tick\ \mathsf{live}\ t \\
& \wedge\ \mathsf{no}\ tock\ \mathsf{at}\ (t-T, t) \wedge \mathsf{no}\ tick\ \mathsf{at}\ (t-1, t) \Rightarrow tock\ \mathsf{live}\ t \\
& \wedge\ tick\ \mathsf{offered}\ t \wedge tock\ \mathsf{offered}\ t \Rightarrow tock\ \mathsf{preferred\ to}\ tick\ @\ t
\end{aligned}$$

Our method of implementing this will be to firstly produce a TCSP process which *nearly* satisfies the above specification: more precisely we will produce a TCSP process all of whose BTCSP refinements satisfy all but the last conjunct of the specification. We will then study which of the refinements also satisfy the final conjunct.

6.3.1 TCSP "implementation"

We seek a TCSP process $CLOCK_0$ all of whose BTCSP refinements satisfy the predicate

$$\begin{aligned} S'(\tau, \sqsubseteq, s) \;\widehat{=}\; & \mathsf{alphabet} \subseteq \{tick, tock\} \\ & \wedge\; tick, tock \;\mathsf{live\ from}\; 0 \\ & \wedge\; tick, tock \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; tick, tock \;\mathsf{at}\; (t, t+1) \\ & \wedge\; tock \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; tock \;\mathsf{at}\; (t, t+T) \\ & \wedge\; \mathsf{no}\; tick, tock \;\mathsf{at}\; (t-1, t) \Rightarrow tick \;\mathsf{live}\; t \\ & \wedge\; \mathsf{no}\; tock \;\mathsf{at}\; (t-T, t) \wedge \mathsf{no}\; tick \;\mathsf{at}\; (t-1, t) \Rightarrow tock \;\mathsf{live}\; t \end{aligned}$$

Using rule 6.2.3 we see that we want a specification $S_0(s, \aleph)$, such that $\Theta S_0 \Rightarrow S'$, and a TCSP process $CLOCK_0$ such that $CLOCK_0$ **sat** $S_0(s, \aleph)$ in $\mathcal{M}_{TF}$. Using the results of section 6.2.1, we see that S_0 can take the obvious form:

$$\begin{aligned} S_0(s, \aleph) \;\widehat{=}\; & \mathsf{alphabet} \subseteq \{tick, tock\} \\ & \wedge\; tick, tock \;\mathsf{live\ from}\; 0 \\ & \wedge\; tick, tock \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; tick, tock \;\mathsf{at}\; (t, t+1) \\ & \wedge\; tock \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; tock \;\mathsf{at}\; (t, t+T) \\ & \wedge\; \mathsf{no}\; tick, tock \;\mathsf{at}\; (t-1, t) \Rightarrow tick \;\mathsf{live}\; t \\ & \wedge\; \mathsf{no}\; tock \;\mathsf{at}\; (t-T, t) \wedge \mathsf{no}\; tick \;\mathsf{at}\; (t-1, t) \Rightarrow tock \;\mathsf{live}\; t \end{aligned}$$

We implement $CLOCK_0$ as the parallel composition of two processes, P and Q. P will ensure that the events are available at the desired intervals; Q will ensure that two events are not available within one second of each other. Recall the TCSP proof rule for parallel composition from [DS89b]:

$$\frac{\begin{array}{l} P \;\mathbf{sat}\; S_P(s, \aleph) \\ Q \;\mathbf{sat}\; S_Q(s, \aleph) \\ S_P(s, \aleph_P) \wedge S_Q(s, \aleph_Q) \Rightarrow S(s, \aleph_P \cup \aleph_Q) \end{array}}{P \parallel Q \;\mathbf{sat}\; S(s, \aleph)}$$

Let

$$\begin{aligned} S_P(s, \aleph) \;\widehat{=}\; & \mathsf{alphabet} \subseteq \{tick, tock\} \\ & \wedge\; tick, tock \;\mathsf{live\ from}\; 0 \\ & \wedge\; tick \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; tick \;\mathsf{at}\; (t, t+1) \\ & \wedge\; tock \;\mathsf{at}\; t \Rightarrow \mathsf{no}\; tock \;\mathsf{at}\; (t, t+T) \\ & \wedge\; \mathsf{no}\; tick \;\mathsf{at}\; (t-1, t) \Rightarrow tick \;\mathsf{live\ from}\; t \\ & \wedge\; \mathsf{no}\; tock \;\mathsf{at}\; (t-T, t) \Rightarrow tock \;\mathsf{live\ from}\; t \end{aligned}$$

$$\begin{aligned} S_Q(s,\aleph) \;\widehat{=}\;& \mathsf{alphabet} \subseteq \{tick, tock\} \\ & \wedge\ tick, tock \ \mathsf{live\ from}\ 0 \\ & \wedge\ tick, tock\ \mathsf{at}\ t \Rightarrow \mathsf{no}\ tick, tock\ \mathsf{at}\ (t, t+1) \\ & \wedge\ \mathsf{no}\ tick, tock\ \mathsf{at}\ (t-1, t) \Rightarrow tick, tock\ \mathsf{live\ from}\ t \end{aligned}$$

Then it is easily seen that $S_P(s, \aleph_P) \wedge S_Q(s, \aleph_Q) \Rightarrow S(s, \aleph_P \cup \aleph_Q)$.

We now seek a process P satisfying S_P. We implement P as an interleaving, $P_1 \quad P_2$; the process P_1 will provide the *ticks*, while P_2 provides the *tocks*. Recall the proof rule for interleaving from [DS89b].

$$\frac{\begin{array}{l} P_1\ \mathbf{sat}\ S_1(s,\aleph) \\ P_2\ \mathbf{sat}\ S_2(s,\aleph) \\ s \in u \quad v \wedge S_1(u,\aleph) \wedge S_2(v,\aleph) \Rightarrow S(s,\aleph) \end{array}}{P \quad Q\ \mathbf{sat}\ S(s,\aleph)}$$

Let

$$\begin{aligned} S_1(s,\aleph) \;\widehat{=}\;& \mathsf{alphabet} \subseteq \{tick\} \\ & \wedge\ tick\ \mathsf{live\ from}\ 0 \\ & \wedge\ tick\ \mathsf{at}\ t \Rightarrow \mathsf{no}\ tick\ \mathsf{at}\ (t, t+1) \\ & \wedge\ \mathsf{no}\ tick\ \mathsf{at}\ (t-1, t) \Rightarrow tick\ \mathsf{live\ from}\ t \end{aligned}$$

$$\begin{aligned} S_2(s,\aleph) \;\widehat{=}\;& \mathsf{alphabet} \subseteq \{tock\} \\ & \wedge\ tock\ \mathsf{live\ from}\ 0 \\ & \wedge\ tock\ \mathsf{at}\ t \Rightarrow \mathsf{no}\ tock\ \mathsf{at}\ (t, t+T) \\ & \wedge\ \mathsf{no}\ tock\ \mathsf{at}\ (t-T, t) \Rightarrow tock\ \mathsf{live\ from}\ t \end{aligned}$$

Then we have $s \in u \quad v \wedge S_1(u,\aleph) \wedge S_2(v,\aleph) \Rightarrow S(s,\aleph)$. It is also an easy exercise to show that $\mu X \quad tick \xrightarrow{1} X\ \mathbf{sat}\ S_1(s,\aleph)$ and $\mu X \quad tock \xrightarrow{T} X\ \mathbf{sat}\ S_2(s,\aleph)$. Hence

$$\mu X \quad tick \xrightarrow{1} X \quad \mu X \quad tock \xrightarrow{T} X\ \mathbf{sat}\ S_P(s,\aleph)$$

We now seek a process Q satisfying S_Q. We will implement Q as a recursion, $\mu X \quad P$. Recall the proof rule for recursion from [DS90]:

$$\frac{X\ \mathbf{sat}\ S(s,\aleph) \Rightarrow P\ \mathbf{sat}\ S(s,\aleph)}{\mu X \quad P\ \mathbf{sat}\ S(s,\aleph)}$$

So we need to find a term P (dependent on X) such that P **sat** $S_Q(s,\aleph)$ whenever X **sat** $S_Q(s,\aleph)$. We implement P as an external choice, $P = P_1 \quad P_2$. The proof rule for external choice is

$$\frac{\begin{array}{l} P_1\ \mathbf{sat}\ S_1(s,\aleph) \\ P_2\ \mathbf{sat}\ S_2(s,\aleph) \\ (S_1(s,\aleph) \vee S_2(s,\aleph)) \wedge S_1(\langle\rangle, \aleph \quad begin\ s) \wedge S_2(\langle\rangle, \aleph \quad begin\ s) \Rightarrow S(s,\aleph) \end{array}}{P_1 \quad P_2\ \mathbf{sat}\ S(s,\aleph)}$$

Let

$$\begin{aligned}
S_1(s,\aleph) \;\widehat{=}\; & \textsf{alphabet} \subseteq \{tick, tock\} \\
& \wedge\ tick \textsf{ live from } 0 \\
& \wedge\ tick, tock \textsf{ at } t \Rightarrow \textsf{no } tick, tock \textsf{ at } (t, t+1) \\
& \wedge\ tick, tock \textsf{ at } [0, t-1] \wedge \textsf{no } tick, tock \textsf{ at } (t-1, t) \Rightarrow tick, tock \textsf{ live from } t \\
S_2(s,\aleph) \;\widehat{=}\; & \textsf{alphabet} \subseteq \{tick, tock\} \\
& \wedge\ tock \textsf{ live from } 0 \\
& \wedge\ tick, tock \textsf{ at } t \Rightarrow \textsf{no } tick, tock \textsf{ at } (t, t+1) \\
& \wedge\ tick, tock \textsf{ at } [0, t-1] \wedge \textsf{no } tick, tock \textsf{ at } (t-1, t) \Rightarrow tick, tock \textsf{ live from } t
\end{aligned}$$

Then it is easy to show that

$$(S_1(s,\aleph) \vee S_2(s,\aleph)) \wedge S_1(\langle\rangle, \aleph \quad begin\ s) \wedge S_2(\langle\rangle, \aleph \quad begin\ s) \Rightarrow S(s,\aleph)$$

Hence it only remains to find processes P_1 and P_2 such that P_1 **sat** $S_1(s,\aleph)$ and P_2 **sat** $S_2(s,\aleph)$ whenever X **sat** $S_Q(S,\aleph)$. Our intuition suggests

$$P_1 \;\widehat{=}\; tick \xrightarrow{1} X \qquad P_2 \;\widehat{=}\; tock \xrightarrow{1} X$$

These definitions can be shown to satisfy the specifications by a simple application of the proof rule for prefixing.

Hence we have shown that the process

$$\begin{aligned}
CLOCK_0 \;\widehat{=}\; & \mu X \quad tick \xrightarrow{1} X \quad \mu X \quad tock \xrightarrow{T} X \\
& \| \\
& \mu X \quad tick \xrightarrow{1} X \quad tock \xrightarrow{1} X
\end{aligned}$$

satisfies the specification $S_0(s,\aleph)$, and so all its BTCSP refinements satisfy the specification $S'(\tau, \sqsubseteq, s)$.

6.3.2 First BTCSP implementation

We seek a BTCSP process $CLOCK$ such that $\varphi_B(CLOCK) = CLOCK_0$ and $CLOCK$ **sat** $S(\tau, \sqsubseteq, s)$ in $\mathcal{M}_{TB}$. We already know that *any* prioritized refinement of $CLOCK_0$ will satisfy all but the last conjunct of S; hence it is enough to find a refinement that satisfies $\hat{S}(\tau, \sqsubseteq, s)$ where

$$\hat{S} \;\widehat{=}\; tick \textsf{ offered } t \wedge tock \textsf{ offered } t \Rightarrow tock \textsf{ preferred to } tick \textsf{ @ } t$$

Our first implementation will make $CLOCK$ the left biased parallel composition of two processes P and Q where

$$\begin{aligned}
\varphi_B(P) &= \mu X \quad tick \xrightarrow{1} X \quad \mu X \quad tock \xrightarrow{1} X \\
\varphi_B(Q) &= \mu X \quad tick \xrightarrow{1} X \quad tock \xrightarrow{1} X
\end{aligned}$$

From the proof rule for left-biased parallel compositiou, given in appendix B.1, we see that we need to find predicates S_P and S_Q for P and Q such that

$$S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \#\!\!\mid \sqsubseteq_Q, s) \qquad (*)$$

We set $S_P = \hat{S}$ and set $S_Q = true$. Then by the definition of parallel composition of offer relations we see that $(*)$ is satisfied. Also Q **sat** S_Q for any Q. Heuce it only remains for us to find P such that P **sat** S_P.

We shall implement P as the right biased interleaving $\mu X \quad tick \xrightarrow{1} X \longrightarrow \mu X \quad tock \xrightarrow{1} X$. Recall from the previous section that $\mu X \quad tick \xrightarrow{1} X$ **sat** alphabet $\subseteq \{tick\}$; hence

$$\mu X \quad tick \xrightarrow{1} X \textbf{ sat } \text{no } tock \text{ offered}$$

Similarly

$$\mu X \quad tock \xrightarrow{T} X \textbf{ sat } \text{no } tick \text{ offered}$$

It is easy to show

$$\mu X \quad tock \xrightarrow{T} X \textbf{ sat } tock \text{ offered } t \Rightarrow tock \text{ preferred to } \{\!|\,|\!\} \text{ @ } t$$

From the proof rule for right biased interleaving we see that we must prove

$$\begin{pmatrix} (\text{no } tock \text{ offered})(\tau, \sqsubseteq_1, \Downarrow_{\sqsubseteq_2, \sqsubseteq_1} s) \\ \wedge\, (\text{no } tick \text{ offered } \wedge (tock \text{ offered } t \Rightarrow tock \text{ preferred to } \{\!|\,|\!\} \text{ @ } t))(\tau, \sqsubseteq_2, \Uparrow_{\sqsubseteq_2, \sqsubseteq_1} s) \end{pmatrix} \Rightarrow$$
$$S_P(\tau, \sqsubseteq_1 \longrightarrow \sqsubseteq_2, s)$$

If $(tick$ offered $t \wedge tock$ offered $t)(\tau, \sqsubseteq_1 \longrightarrow \sqsubseteq_2, s)$ and (no $tock$ offered)$(\tau, \sqsubseteq_1, \Downarrow_{\sqsubseteq_2, \sqsubseteq_1} s)$ and (no $tick$ offered)$(\tau, \sqsubseteq_2, \Uparrow_{\sqsubseteq_2, \sqsubseteq_1} s)$ then we must have that $(tick$ offered $t)(\tau, \sqsubseteq_1, \Downarrow_{\sqsubseteq_2, \sqsubseteq_1} s)$ and $(tock$ offered $t)(\tau, \sqsubseteq_2, \Uparrow_{\sqsubseteq_2, \sqsubseteq_1} s)$. Thus $(tock$ preferred to $\{\!|\,|\!\}$ @ $t)(\tau, \sqsubseteq_2, \Uparrow_{\sqsubseteq_2, \sqsubseteq_1} s)$. Hence by the definition of interleaving of offer relations, $(tock$ preferred to $tick$ @ $t)(\tau, \sqsubseteq_1 \longrightarrow \sqsubseteq_2, s)$, as required. Hence

$$\mu X \quad tick \xrightarrow{1} X \longrightarrow \mu X \quad tock \xrightarrow{1} X \textbf{ sat}$$
$$tick \text{ offered } t \wedge tock \text{ offered } t \Rightarrow tock \text{ preferred to } tick \text{ @ } t$$

Thus, we have shown that both of the processes

$$(\mu X \quad tick \xrightarrow{1} X \longrightarrow \mu X \quad tock \xrightarrow{T} X) \#\!\!\mid (\mu X \quad tick \xrightarrow{1} X \sqsubset\!\!\!\square\; tock \xrightarrow{1} X)$$
and
$$(\mu X \quad tick \xrightarrow{1} X \longrightarrow \mu X \quad tock \xrightarrow{T} X) \#\!\!\mid (\mu X \quad tick \xrightarrow{1} X \;\square\!\!\!\sqsupset tock \xrightarrow{1} X)$$

satisfy our original predicate S.

6.3.3 Second BTCSP implementation

We will now try to implement the clock using a right biased parallel composition of processes P and Q such that

$$
\begin{aligned}
\varphi_B(P) &= \mu X \quad tick \xrightarrow{1} X \quad \mu X \quad tock \xrightarrow{1} X \\
\varphi_B(Q) &= \mu X \quad tick \xrightarrow{1} X \quad tock \xrightarrow{1} X
\end{aligned}
$$

Examining the proof rule for right biased parallel composition we see that we must find predicates S_P and S_Q for P and Q such that

$$S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow \hat{S}(\tau, \sqsubseteq_P \mathbin{\|\!\!\succ} \sqsubseteq_Q, s)$$

We instantiate S_P with *true* and S_Q with $\hat{S}$. As in the previous subsection, it only remains for us to find a process Q satisfying S_Q.

Following the results of section 6.3.1 we implement Q as a recursion $\mu X \quad Q'$ such that $\varphi_B(Q') = tick \xrightarrow{1} X \quad tock \xrightarrow{1} X$, and if X **sat** $\hat{S}$ then Q' **sat** $\hat{S}$. We then implement Q' as the right biased choice $tick \xrightarrow{1} X \boxdot tock \xrightarrow{1} X$. We seek specifications S'_P for $tick \xrightarrow{1} X$ and S'_Q for $tock \xrightarrow{1} X$ that allow us to prove that $tick \xrightarrow{1} X \boxdot tock \xrightarrow{1} X$ **sat** $\hat{S}$. We instantiate S'_P and S'_Q by

$$
\begin{aligned}
S'_P \;\widehat{=}\;& tick \text{ preferred to } \{\!|\,|\!\} \text{ from } 0 \\
& \wedge\, tick \text{ offered } t \wedge tock \text{ offered } t \Rightarrow tock \text{ preferred to } tick \text{ @ } t
\end{aligned}
$$

$$
\begin{aligned}
S'_Q \;\widehat{=}\;& tock \text{ preferred to } \{\!|\,|\!\} \text{ from } 0 \\
& \wedge\, tick \text{ offered } t \wedge tock \text{ offered } t \Rightarrow tock \text{ preferred to } tick \text{ @ } t
\end{aligned}
$$

From the proof rule for right-biased choice we see that we have the following proof obligations:

$$
\begin{aligned}
& S'_P(\tau, \sqsubseteq_P, \prec\succ) \wedge S'_Q(\tau, \sqsubseteq_Q, \prec\succ) \Rightarrow \hat{S}(\tau, \sqsubseteq_P \boxdot \sqsubseteq_Q, \prec\succ) \\
& s \neq \prec\succ \wedge\, begin\, s = t \wedge S'_P(\tau, \sqsubseteq_P, s) \wedge S'_Q(t, \sqsubseteq_Q, \prec\succ) \wedge s \uparrow t \not\sqsupset_Q (t, \{\!|\,|\!\}) \Rightarrow \hat{S}(\tau, \sqsubseteq_P \boxdot \sqsubseteq_Q, s) \\
& \begin{pmatrix} s \neq \prec\succ \wedge\, begin\, s = t \wedge S'_P(t, \sqsubseteq_P, \prec\succ) \wedge S'_Q(\tau, \sqsubseteq_Q, s) \\ \wedge\, (s \uparrow t \sqsupset_Q (t, \{\!|\,|\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_P) \end{pmatrix} \Rightarrow \hat{S}(\tau, \sqsubseteq_P \boxdot \sqsubseteq_Q, s)
\end{aligned}
$$

These are easily proven using the definition of right biased choice of offer relations.

It remains to show that $tick \xrightarrow{1} X$ **sat** S'_P and $tock \xrightarrow{1} X$ **sat** S'_Q. We prove the former result; the latter is identical. From the proof rule for prefixing, remembering that X **sat** $\hat{S}$, we see that we have the following proof obligations:

$$
\begin{aligned}
& S'_P(\tau, [0, \tau] \otimes \langle \{\!|tick|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \\
& t' \quad \tau < t' + 1 \Rightarrow S'_P(\tau, [0, t'] \otimes \langle \{\!|tick|\!\}, \{\!|\,|\!\} \rangle \quad (t', \tau] \oslash \langle \{\!|\,|\!\} \rangle . \prec (t', tick) \succ) \\
& \hat{S}(\tau - 1 - t', \sqsubseteq, s) \wedge \tau \quad t' + 1 \Rightarrow \\
& \quad S'_P(\tau, [0, t'] \otimes \langle \{\!|tick|\!\}, \{\!|\,|\!\} \rangle \quad (t', t' + 1) \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq + t' + 1, (t', tick) \quad s + t' + 1)
\end{aligned}
$$

These can be proved by careful checking. Hence $tick \xrightarrow{1} X$ **sat** S'_P, and so we have shown that both of the processes

$$(\mu X \quad tick \xrightarrow{1} X \longleftarrow \mu X \quad tock \xrightarrow{T} X) \interleave (\mu X \quad tick \xrightarrow{1} X \mathrel{\Box} tock \xrightarrow{1} X)$$

and

$$(\mu X \quad tick \xrightarrow{1} X \longrightarrow \mu X \quad tock \xrightarrow{T} X) \interleave (\mu X \quad tick \xrightarrow{1} X \mathrel{\Box} tock \xrightarrow{1} X)$$

satisfy our original predicate S.

Chapter 7

Specification and Proof of Probabilistic Processes

In chapter 5 we developed proof rules that could be used for proving that a probabilistic process satisfies an unprobabilistic specification, i.e. a specification that is supposed to hold of *all* behaviours of a process. Our proof rules allowed us to translate a specification on a composite process into specifications on its subcomponents. In this chapter we aim to extend the proof system so that it can deal with *probabilistic* specifications.

In section 7.1 we will describe the form of our specifications: we will write $P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S(\tau, \sqsubseteq, s)$ to specify that, whatever the environment offers, the probability that process P performs a behaviour $(\tau, \sqsubseteq, s)$ that satisfies the predicate $S(\tau, \sqsubseteq, s)$ is at least p. We will also define conditional probabilities: we will write $P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S(\tau, \sqsubseteq, s) \mid G(\tau, \sqsubseteq, s)$ to specify that the probability that P performs a behaviour that satisfies S *given that* it satisfies G is at least p. We will present a number of proof rules which are independent of the syntax of our language.

In section 7.2 we will explain, via a number of examples, why proving specifications for probabilistic specifications can be considerably harder than in the unprobabilistic case: a number of factors introduce difficulties not present in the unprobabilistic case. We will show how to produce proof rules that overcome these difficulties. In section 7.3 we derive proof rules for all the constructs of the language.

In section 7.4 we present a large case study. We describe a protocol transmitting messages over an unreliable medium. We show that it acts like a buffer, and perform an analysis of its performance: we prove a result that gives the probability of a message being correctly transmitted within a certain amount of time.

7.1 Specification of probabilistic processes

In this section we introduce the form of our probabilistic specifications and give a few basic rules for manipulating them which are independent of the syntax of the language. We begin by considering the basic specification statement; we then go on to consider conditional probabilities.

7.1.1 The basic specification statement

We will write P $\mathbf{sat}_\rho^{\geqslant p}$ $S(\tau, \sqsubseteq, s)$ to mean that in all environments the probability of P performing a behaviour $(\tau, \sqsubseteq, s)$ that satisfies the predicate S is at least p. To define this formally we want to be able to discuss the probability of a process P satisfying some behavioural specification $S(\tau, \sqsubseteq, s)$ in a given environment Ω and with variable binding ρ; we will write this as ${}^{\Omega,\rho}_{P}$ $S(\tau, \sqsubseteq, s)$. Recall that we allow the environment to be a function of the offer relation of a process: we write $\Omega(\sqsubseteq)$ when we want to stress this.

Definition 7.1.1 (Probability of satisfaction) If $P \in PBTCSP$, $\Omega \in OFFREL \to EOFF$, $\rho \in ENV$, and $S \in BEH \to Bool$, then

$$
{}^{\Omega,\rho}_{P} \; S(\tau, \sqsubseteq, s) \;\widehat{=}\; \sum \left\{\!\left| \mathcal{P}_{PBT} \; P \; \rho(\tau, \sqsubseteq, s) \mid S(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s) \; compat \; \Omega(\sqsubseteq) \right|\!\right\}
$$

$\diamondsuit$

${}^{\Omega,\rho}_{P}$ $S(\tau, \sqsubseteq, s)$ is the probability, given variable binding ρ, of P performing a behaviour that is compatible with Ω and that satisfies S. We will drop the P, the Ω, the ρ, and the argument $(\tau, \sqsubseteq, s)$ of a predicate where this will not cause confusion.

We can now formally define our specification statement:

Definition 7.1.2 (Probabilistic satisfaction) If $P \in PBTCSP$, $\rho \in ENV$, $S \in BEH \to Bool$, and $p \in [0, 1]$, then

$$
P \; \mathbf{sat}_\rho^{\geqslant p} \; S(\tau, \sqsubseteq, s) \Leftrightarrow \forall \Omega : EOFF \quad {}^{\Omega,\rho}_{P} \; S(\tau, \sqsubseteq, s) \qquad p
$$

$\diamondsuit$

P $\mathbf{sat}_\rho^{\geqslant p}$ $S(\tau, \sqsubseteq, s)$ if in all environments Ω the probability that P performs a behaviour that satisfies S is at least p. If P is a process (as opposed to a term) then its semantic value is independent of the variable binding so in this case it makes sense to omit reference to the variable binding and to write P $\mathbf{sat}^{\geqslant p}$ $S(\tau, \sqsubseteq, s)$. We will also drop the argument $(\tau, \sqsubseteq, s)$ of S where this will not cause confusion.

7.1.2 Conditional specifications

We will sometimes want to say that with some probability a process satisfies some specification S *given that* it satisfies some other specification G.

Definition 7.1.3 (Conditional satisfaction) If $P \in PBTCSP$, S and G are predicates, $\rho \in ENV$, and $p \in [0, 1]$ then

$$
\begin{aligned}
&P \; \mathbf{sat}_\rho^{\geqslant p} \; S(\tau, \sqsubseteq, s) \mid G(\tau, \sqsubseteq, s) \Leftrightarrow \\
&\quad \forall \Omega : EOFF \quad {}^{\Omega,\rho}_{P} \; S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \qquad p. \; {}^{\Omega,\rho}_{P} \; G(\tau, \sqsubseteq, s)
\end{aligned}
$$

$\diamondsuit$

In tbe case where ${}_{P}^{\Omega,\rho}\ G(\tau,\sqsubseteq,s)\ > 0$ this reduces to the more familiar

$$\frac{{}_{P}^{\Omega,\rho}\ S(\tau,\sqsubseteq,s) \wedge G(\tau,\sqsubseteq,s)}{{}_{P}^{\Omega,\rho}\ G(\tau,\sqsubseteq,s)} \quad p$$

We sball normally adopt tbe convention of writing G, G' etc. for the *G*iven predicate.

The reader should note tbat conditional specification is different to specification of an implication, i.e. $P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S(\tau,\sqsubseteq,s) \mid G(\tau,\sqsubseteq,s)$ is not the same as $P\ \mathbf{sat}_{\rho}^{\geqslant p}\ G(\tau,\sqsubseteq,s) \Rightarrow S(\tau,\sqsubseteq,s)$. Consider the process $P \mathrel{\hat{=}} (a \longrightarrow (b\ {}_{p}\sqcap_{q}\ c))\ {}_{p'}\sqcap_{q'}\ STOP$. Let S_a be the specification that an a is performed; let S_b be the specification that a b is offered. Then $P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S_b \mid S_a$ (but it doesn't satisfy this with any higher probability) wbile $P\ \mathbf{sat}_{\rho}^{\geqslant pp'+q'}\ S_a \Rightarrow S_b$. In section 7.1.4 we will give some rules relating these two concepts.

In the following sections we give a number of proof rules for probabilistic specifications that are independent of the syntax of the language.

7.1.3 Basic proof rules for probabilistic specifications

If a process satisfies a specification with some probability, then it certainly satisfies that specification with any lower probability.

Rule 7.1.4 (Lower probabilities)

$$\frac{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \mid G}{P\ \mathbf{sat}_{\rho}^{\geqslant q}\ S \mid G}\left[p \quad q\right]$$

$\triangle$

Every process obeys every predicate witb probability at least zero.

Rule 7.1.5 (Zero probability)

$$\frac{}{P\ \mathbf{sat}_{\rho}^{\geqslant 0}\ S \mid G}$$

$\triangle$

The following rule allows us to weaken the given predicate and strengthen the conjunct of the two predicates.

Rule 7.1.6 (Weaken and strengthen specifications)

$$\frac{\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S' \mid G' \\ S'(\tau,\sqsubseteq,s) \wedge G'(\tau,\sqsubseteq,s) \Rightarrow S(\tau,\sqsubseteq,s) \wedge G(\tau,\sqsubseteq,s) \\ G(\tau,\sqsubseteq,s) \Rightarrow G'(\tau,\sqsubseteq,s) \end{array}}{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \mid G}$$

$\triangle$

The following rule can be derived from the above by taking $G = G'$.

Rule 7.1.7 (Weaken specifications)

$$\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geq p}\ S' \mid G \\ S'(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \\ \hline P\ \mathbf{sat}_{\rho}^{\geq p}\ S \mid G \end{array}$$

$\triangle$

A process satisfies specifications S and G whenever it satisfies G and $S \mid G$.

Rule 7.1.8 (Conjunction of specifications)

$$\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geq p}\ G \\ P\ \mathbf{sat}_{\rho}^{\geq q}\ S \mid G \\ \hline P\ \mathbf{sat}_{\rho}^{\geq pq}\ S \wedge G \end{array}$$

$\triangle$

The following is an easy corollary of this:

Rule 7.1.9:

$$\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geq p}\ G \\ P\ \mathbf{sat}_{\rho}^{\geq q}\ S \mid G \\ \hline P\ \mathbf{sat}_{\rho}^{\geq pq}\ S \end{array}$$

$\triangle$

7.1.4 Relating conditional and unconditional specifications

The following rule shows that the specifications $S \mid true$ and S are equivalent:

Rule 7.1.10:

$$\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geq p}\ S \\ \hline P\ \mathbf{sat}_{\rho}^{\geq p}\ S \mid true \end{array} \qquad\qquad \begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geq p}\ S \mid true \\ \hline P\ \mathbf{sat}_{\rho}^{\geq p}\ S \end{array}$$

$\triangle$

Proof: This follows from the law of PBTCSP that states that in any environment the sum of the probabilities of all possible behaviours is one, i.e. $\underset{P}{\overset{\Omega,\rho}{}}\ true\ = 1$. $\square$

This rule can be used to adapt many of the other rules so as to apply them to unconditional specifications. For example taking $G = G' = true$ in rule 7.1.7 we get the rule

Rule 7.1.11:

$$\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geq p}\ S' \\ S'(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \\ \hline P\ \mathbf{sat}_{\rho}^{\geq p}\ S \end{array}$$

$\triangle$

We can relate conditional specification to specification of an implication:

Rule 7.1.12:

$$\frac{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \mid G}{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ G \Rightarrow S}$$

$\triangle$

Proof: It is enough to show that

$$G \Rightarrow S \qquad \frac{S \wedge G}{G}$$

which follows easily via algebraic manipulations. $\square$

In the case where $p = 1$ we also have the converse:

Rule 7.1.13:

$$\frac{P\ \mathbf{sat}_{\rho}\ G \Rightarrow S\,.}{P\ \mathbf{sat}_{\rho}^{\geqslant 1}\ S \mid G}$$

$\triangle$

Proof: Suppose $P\ \mathbf{sat}_{\rho}\ G \Rightarrow S$. Then for all environments Ω:

$$\begin{aligned}
& {}^{\Omega,\rho}_{P}\ S \wedge G \\
= & \left\langle \text{definition} \right\rangle \\
& \textstyle\sum \left\{\!\left| \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \right|\!\right\} \\
= & \left\langle P\ \mathbf{sat}_{\rho}\ G \Rightarrow S \right\rangle \\
& \textstyle\sum \left\{\!\left| \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid G(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \right|\!\right\} \\
= & \left\langle \text{definition} \right\rangle \\
& {}^{\Omega,\rho}_{P}\ G
\end{aligned}$$

Hence ${}^{\Omega,\rho}_{P}\ S \wedge G \quad 1 \times {}^{\Omega,\rho}_{P}\ G$ so $P\ \mathbf{sat}_{\rho}^{\geqslant 1}\ S \mid G$. $\square$

If a process always satisfies some predicate, then it satisfies it with probability one.

Rule 7.1.14 (Certainty)

$$\frac{P\ \mathbf{sat}_{\rho}\ S}{P\ \mathbf{sat}_{\rho}^{\geqslant 1}\ S}$$

$\triangle$

Proof: This follows by taking $G = \mathit{true}$ in the previous rule, and making use of rule 7.1.10.
□

If we know that *all* behaviours of a process satisfy a particular predicate, then we can add this predicate to a probabilistic specificatiou without affecting its truth:

Rule 7.1.15:

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S \\ P\ \mathbf{sat}_\rho^{\geqslant p}\ S' \mid G \end{array}}{P\ \mathbf{sat}_\rho^{\geqslant p}\ S \wedge S' \mid G}$$

△

Proof: Assume the premises; then for all environments Ω,

$$\begin{array}{ll} & {}^{\Omega,\rho}_{\rho}\ S \wedge S' \wedge G \\ = & \langle \text{definition of } \ \rangle \\ & \sum\{\!\!\{ \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \wedge S(\tau, \sqsubseteq, s) \wedge S'(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \}\!\!\} \\ = & \left\langle \begin{array}{l} \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) > 0 \Rightarrow (\tau, \sqsubseteq, s) \in \mathcal{A}_{PBT}\ P\ \rho \Rightarrow S(\tau, \sqsubseteq, s) \\ \text{by axiom P4 of the semantic space and premise 1} \end{array} \right\rangle \\ & \sum\{\!\!\{ \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \wedge S'(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \}\!\!\} \\ & \langle \text{premise } 2 \rangle \\ & p \cdot\ {}^{\Omega,\rho}_{P}\ G \end{array}$$

So $P\ \mathbf{sat}_\rho^{\geqslant p}\ S \wedge S' \mid G$. □

This is a particularly useful rule: often in proving a probabilistic result one begins by proving a number of lemmas that do not involve probabilities: one proves properties that hold of *all* behaviours of a process. This rule means that we can make use of the lemmas by adding their results to any probabilistic results we can prove about the processes: for example, if we know that all behaviours of P satisfy S, and we want to make use of the fact that $P\ \mathbf{sat}_\rho^{\geqslant p}\ S \wedge S' \mid G$ then it is enough to prove $P\ \mathbf{sat}_\rho^{\geqslant p}\ S' \mid G$. In section 7.4 we will consider a protocol: we will begin by proving that it acts like a one place buffer; in doing this we will prove a number of results, for example about the order in which events are performed, that will prove useful when we consider the probabilistic aspects of the protocol.

7.1.5 Simplifying conditional specifications

The following two rules allow us to simplify conditional specifications. It will often be the case that the left hand side of a conditional specification is of the form $E \Rightarrow S$ for some environmental condition E, and the right hand side also has a conjunct depending on E, i.e. of the form $E \Rightarrow G$. In this case we can drop the E from the right hand side.

Rule 7.1.16:

$$\frac{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ E \Rightarrow S \mid G \wedge G'}{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ E \Rightarrow S \mid (E \Rightarrow G) \wedge G'}$$

△

Note that not all of the right hand side has to depend upon the environmental condition E.

Proof: It is enough to show that,

$$\frac{(E \Rightarrow S) \wedge G \wedge G'}{G \wedge G'} \qquad \frac{(E \Rightarrow S) \wedge (E \Rightarrow G) \wedge G'}{(E \Rightarrow G) \wedge G'}$$

which can be easily proved by algebraic manipulations. □

If a conjunct appears on both sides of a conditional specification then we can drop it from the right hand side.

Rule 7.1.17:

$$\frac{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \wedge S' \mid G}{P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \wedge S' \mid G \wedge S'}$$

△

Proof: Assume the premise of the proof rule. Then we have

$$\begin{array}{ll} & (S \wedge S') \wedge (G \wedge S') \\ = & \left\langle \text{predicate calculus} \right\rangle \\ & (S \wedge S') \wedge G \\ & \left\langle \text{premise} \right\rangle \\ & p. \quad G \\ & \left\langle \text{predicate calculus} \right\rangle \\ & p. \quad G \wedge S' \end{array}$$

So $P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \wedge S' \mid G \wedge S'$. □

7.1.6 Disjoint specifications

We define a vector of predicates to be disjoint if no two of them can be true at the same time:

Definition 7.1.18 (Disjointness of specifications) For all $i \in I$ let S_i be a predicate of type $X \to \mathit{Bool}$; then we say that the vector of predicates $\langle S_i(x) \mid i \in I \rangle$ is *disjoint* iff

$$i, j : I\ ;\ x : X \quad i \neq j \wedge S_i(x) \wedge S_j(x)$$

◇

We use this definition in the following rule:

Rule 7.1.19 (Disjoint specifications)

$$\frac{\begin{array}{l}\forall\, i : I \;\; P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i \mid G_i \\ \forall\, i : I \;\; S_i(\tau, \sqsubseteq, s) \wedge G_i(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\ \forall\, i : I \;\; G(\tau, \sqsubseteq, s) \Rightarrow G_i(\tau, \sqsubseteq, s)\end{array}}{P\ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i p_i}\ S \mid G}\left[\,\langle \hat{S}_i(\tau, \sqsubseteq, s) \mid i \in I\rangle\ \mathit{disjoint}\,\right]$$

where $\hat{S}_i(\tau, \sqsubseteq, s) \cong S_i(\tau, \sqsubseteq, s) \wedge G_i(\tau, \sqsubseteq, s)$. $\triangle$

The rule allows us to add the probabilities of a number of disjoint specifications to obtain the probability of one of them occurring.

Proof: Assume the premises of the proof rule. Then for any environment Ω we have

$$\begin{array}{ll}
 & {}_{P}^{\Omega,\rho}\ S \wedge G \\
= & \left\langle \text{definition of } \right\rangle \\
 & \sum\left\{\!\!\left|\, \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \,\right|\!\!\right\} \\
 & \left\langle \text{partitioning using the side condition} \right\rangle \\
 & \sum\left\{\!\!\left|\, \sum\left\{\!\!\left|\, \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \,\middle|\, \begin{array}{l} S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \wedge S_i(\tau, \sqsubseteq, s) \\ \wedge\ G_i(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \end{array} \right|\!\!\right\} \,\middle|\, i \in I \right|\!\!\right\} \\
= & \left\langle \text{premise } 2 \right\rangle \\
 & \sum\left\{\!\!\left|\, \sum\left\{\!\!\left|\, \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid S_i(\tau, \sqsubseteq, s) \wedge G_i(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \,\right|\!\!\right\} \mid i \in I \right|\!\!\right\} \\
 & \left\langle \text{premise } 1 \right\rangle \\
 & \sum\left\{\!\!\left|\, p_i \cdot \sum\left\{\!\!\left|\, \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid G_i(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \,\right|\!\!\right\} \mid i \in I \right|\!\!\right\} \\
 & \left\langle \text{premise } 3 \right\rangle \\
 & \sum\left\{\!\!\left|\, p_i \cdot \sum\left\{\!\!\left|\, \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid G(\tau, \sqsubseteq, s) \wedge (\tau, \sqsubseteq, s)\ \mathit{compat}\ \Omega \,\right|\!\!\right\} \mid i \in I \right|\!\!\right\} \\
= & \left\langle \text{rearranging; definition of } \right\rangle \\
 & \sum\left\{\!\!\left|\, p_i \mid i \in I \,\right|\!\!\right\} \cdot\ {}_{P}^{\Omega,\rho}\ G
\end{array}$$

Hence $P\ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i p_i}\ S \mid G$. $\square$

The following rule is an easy corollary of this:

Rule 7.1.20 (Disjunction)

$$\frac{\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \\ P\ \mathbf{sat}_{\rho}^{\geqslant q}\ S' \end{array}}{P\ \mathbf{sat}_{\rho}^{\geqslant p+q}\ S \vee S'}\left[\,\langle S(\tau, \sqsubseteq, s), S'(\tau, \sqsubseteq, s)\rangle\ \mathit{disjoint}\,\right]$$

$\triangle$

7.1.7 Inductive proof rules

We bave an inductive principle for our specifications:

Rule 7.1.21:

$$\begin{array}{l} P\,\mathbf{sat}_\rho^{\geqslant p}\ S_0 \mid G \\ \underline{\forall m: \quad P\,\mathbf{sat}_\rho^{\geqslant q}\ S_{m+1} \mid S_m \wedge G} \\ P\,\mathbf{sat}_\rho^{\geqslant p\cdot q^n}\ S_n \mid G \end{array}$$

$\triangle$

Proof: By numerical induction on n. The base case follows immediately from the first premise. For the inductive step,

$$\begin{array}{l} \quad S_{n+1} \wedge G \\ \left\langle \text{strengthening predicate} \right\rangle \\ \quad S_{n+1} \wedge (S_n \wedge G) \\ \left\langle \text{premise 2} \right\rangle \\ q. \quad S_n \wedge G \\ \left\langle \text{inductive hypotbesis} \right\rangle \\ q.p.q^n. \quad G \end{array}$$

So $P\,\mathbf{sat}_\rho^{\geqslant p.q^{n+1}}\ S_{n+1} \mid G$. $\square$

The following version or tbe induction rule will prove useful:

Rule 7.1.22:

$$\begin{array}{l} P\,\mathbf{sat}_\rho^{\geqslant p'}\ S_0 \mid G \\ \forall m: \quad P\,\mathbf{sat}_\rho^{\geqslant q}\ S_{m+1} \mid S_m \wedge G \\ \underline{P\,\mathbf{sat}_\rho^{\geqslant p}\ S'_{n+1} \mid S_n \wedge G} \\ P\,\mathbf{sat}_\rho^{\geqslant p'.p.q^n}\ S'_{n+1} \mid G \end{array}$$

$\triangle$

This will be used as follows: G will represent some initial state; the S'_ns will represent some 'desirable states'; the S_ns will represent states from which it may still be possible to reach a desirable state. The rule then gives the probability of a desirable state being reached. This is illustrated in figure 7.1.

Proof: Using the first two premises and tbe previous proof rule we bave tbat $P\,\mathbf{sat}_\rho^{\geqslant p'.q^n}$ $S_n \mid G$. Then as above, using the third premise,

$$S'_{n+1} \wedge G \qquad S'_{n+1} \wedge (S_n \wedge G) \qquad p. \quad S_n \wedge G \qquad p.p'.q^n. \quad G$$

So $P\,\mathbf{sat}_\rho^{\geqslant p'.p.q^n}\ S'_{n+1} \mid G$. $\square$

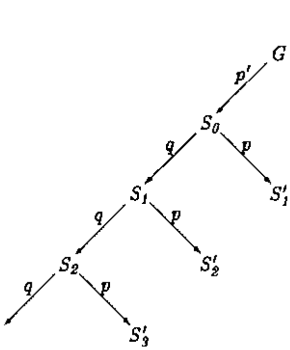

Figure 7.1: Representation of rule 7.1.22

In section 7.4 we will apply this to a protocol transmitting over a medium that correctly transmits messages with probability p. G will represent the state where an input is received; S_n will represent the state where it tries transmitting for the $n+1$th time. With probability p the message is correctly transmitted, which is represented by state S'_{n+1}; with probability q, the message is not correctly transmitted and the protocol will try retransmitting, i.e. it will go into state S_{n+1}. The rule then gives the probability of the message being correctly transmitted at the $n+1$th attempt.

7.2 Complications with probabilistic proofs

The reader may not be surprised to find that proving probabilistic specifications of processes is considerably harder than proving unprobabilistic specifications: there are a number of complications which make the proof rules more difficult to use. In this section we give a number of examples which demonstrate these complications and show how they can be overcome.

Recall the proof rule for proving that an unprobabilistic specification holds for a parallel composition:

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho\; S_P \\ Q \;\mathbf{sat}_\rho\; S_Q \\ s \subseteq A \cup B \wedge S_P(\tau, \sqsubseteq_P, s \;\; A) \wedge S_Q(\tau, \sqsubseteq_Q, s \;\; B) \Rightarrow S(\tau, \sqsubseteq_P {}^{A}\|^{B} \sqsubseteq_Q, s) \end{array}}{P \;{}^{A}\|^{B}\; Q \;\mathbf{sat}_\rho\; S}$$

By analogy with this, we would expect the following proof rule for probabilistic specifications to hold

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho^{\geq p}\; S_P \\ Q \;\mathbf{sat}_\rho^{\geq q}\; S_Q \\ s \subseteq A \cup B \wedge S_P(\tau, \sqsubseteq_P, s \;\; A) \wedge S_Q(\tau, \sqsubseteq_Q, s \;\; B) \Rightarrow S(\tau, \sqsubseteq_P {}^{A}\|^{B} \sqsubseteq_Q, s) \end{array}}{P \;{}^{A}\|^{B}\; Q \;\mathbf{sat}_\rho^{\geq p \cdot q}\; S}$$

$P\ {}^{A}\Vert^{B}\ Q$ satisfies some specification with probability $p.q$ if P and Q satisfy corresponding specifications with probabilities p and q. This rule is indeed true. However, we shall see that this rule is not strong enough for all our purposes.

7.2.1 Conditional specifications

Consider first of all conditional specifications. We would like to be able to reduce a conditional specification on a parallel composition to conditional specifications on the subcomponents. The following rule does this for us:

$$
\frac{
\begin{array}{l}
P\ \mathbf{sat}_p^{\geqslant p}\ S_P \mid G_P \\
Q\ \mathbf{sat}_p^{\geqslant q}\ S_Q \mid G_Q \\
s \subseteq A \cup B \wedge S_P(\tau, \sqsubseteq_P, s\ \ A) \wedge G_P(\tau, \sqsubseteq_P, s\ \ A) \wedge S_Q(\tau, \sqsubseteq_Q, s\ \ B) \wedge G_Q(\tau, \sqsubseteq_Q, s\ \ B) \\
\quad \Rightarrow S(\tau, \sqsubseteq_P {}^{A}\Vert^{B} \sqsubseteq_Q, s) \wedge G(\tau, \sqsubseteq_P {}^{A}\Vert^{B} \sqsubseteq_Q, s) \\
s \subseteq A \cup B \wedge G(\tau, \sqsubseteq_P {}^{A}\Vert^{B} \sqsubseteq_Q, s) \Rightarrow G_P(\tau, \sqsubseteq_P, s\ \ A) \wedge G_Q(\tau, \sqsubseteq_Q, s\ \ B)
\end{array}
}{
P\ {}^{A}\Vert^{B}\ Q\ \mathbf{sat}_p^{\geqslant pq}\ S \mid G
}
$$

Informally, if G holds of a behaviour of $P\ {}^{A}\Vert^{B}\ Q$ then premise 4 tells us that G_P and G_Q hold of the corresponding behaviours of P and Q. Premises 1 and 2 then tell us that with probability p, the behaviour of P satisfies S_P and G_P, and with probability q, the behaviour of Q satisfies S_Q and G_Q. Premise 3 is then enough to deduce that S and G hold of $P\ {}^{A}\Vert^{B}\ Q$'s behavionr.

The following slightly simpler rule is an immediate corollary of this:

$$
\frac{
\begin{array}{l}
P\ \mathbf{sat}_p^{\geqslant p}\ S_P \mid G_P \\
Q\ \mathbf{sat}_p^{\geqslant q}\ S_Q \mid G_Q \\
s \subseteq A \cup B \wedge S_P(\tau, \sqsubseteq_P, s\ \ A) \wedge S_Q(\tau, \sqsubseteq_Q, s\ \ B) \Rightarrow S(\tau, \sqsubseteq_P {}^{A}\Vert^{B} \sqsubseteq_Q, s) \\
s \subseteq A \cup B \wedge G(\tau, \sqsubseteq_P {}^{A}\Vert^{B} \sqsubseteq_Q, s) \Leftrightarrow G_P(\tau, \sqsubseteq_P, s\ \ A) \wedge G_Q(\tau, \sqsubseteq_Q, s\ \ B)
\end{array}
}{
P\ {}^{A}\Vert^{B}\ Q\ \mathbf{sat}_p^{\geqslant pq}\ S \mid G
}
$$

7.2.2 Multiple possibilities

Consider the process $P \Vert Q$ where $P \mathrel{\widehat{=}} a\ {}_{0.3}\sqcap_{0.7}\ b$ and $Q \mathrel{\widehat{=}} a\ {}_{0.6}\sqcap_{0.4}\ b$. We would like to be able to be able to prove that this deadlocks immediately with probability $0.3 \times 0.4 + 0.7 \times 0.6 = 0.54$; i.e. $P \Vert Q\ \mathbf{sat}_p^{\geqslant 0.54}$ silent where $\mathsf{silent}(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} s = \prec\succ$. However, there are not predicates S_P and S_Q that allow this to be proved using the above rule. The reason for this is that a deadlocked behaviour can come about in two ways: either from P offering a aud Q offering b, or vice versa.

The following proof rule meets our requirements: a proof obligatiou on $P\ {}^{A}\Vert^{B}\ Q$ is reduced to a number of proof obligations on the subcomponents.

$$
\frac{
\begin{array}{ll}
\forall i : I & P\ \mathbf{sat}_p^{\geqslant p_i}\ S_{P,i} \\
\forall i : I & Q\ \mathbf{sat}_p^{\geqslant q_i}\ S_{Q,i} \\
\forall i : I & s \subseteq A \cup B \wedge S_{P,i}(\tau, \sqsubseteq_P, s\ \ A) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s\ \ B) \Rightarrow \\
& \qquad S(\tau, \sqsubseteq_P {}^{A}\Vert^{B} \sqsubseteq_Q, s)
\end{array}
}{
P\ {}^{A}\Vert^{B}\ Q\ \mathbf{sat}_p^{\geqslant \Sigma_i\, p_i q_i}\ S
}
\quad
\left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ \textit{disjoint} \end{array}\right]
$$

where

$$\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\widehat{=}} s \subseteq A \cup B \wedge S_{P,i}(\tau, \sqsubseteq_P, s \quad A) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s \quad B)$$

Any pair of specifications $S_{P,i}$ and $S_{Q,i}$ is enough to ensure that S holds of $P \;{}_A\|_B\; Q$. Note that our original proof rule is a special case of this where I is a singleton set. We need to avoid double counting: hence we need the side condition, which ensures that we never consider the same pair of behaviours (for P and Q) twice.

We illustrate this proof rule by applying it to our example. Define the predicate only offers by

$$\text{only offers } c \mathrel{\widehat{=}} c \text{ live from } 0 \wedge \text{no } \Sigma \setminus c \text{ offered}$$

only offers c is the predicate that specifies that the process is only willing to perform the event c. Let

$$\begin{array}{llll} S_{P,1} \mathrel{\widehat{=}} \text{only offers } a & & p_1 \mathrel{\widehat{=}} 0.3 \\ S_{P,2} \mathrel{\widehat{=}} \text{only offers } b & & p_2 \mathrel{\widehat{=}} 0.7 \\ S_{Q,1} \mathrel{\widehat{=}} \text{only offers } b & & q_1 \mathrel{\widehat{=}} 0.4 \\ S_{Q,2} \mathrel{\widehat{=}} \text{only offers } a & & q_2 \mathrel{\widehat{=}} 0.6 \end{array}$$

Then it is easily seen that

$$\forall i \in \{1, 2\} \quad P \ \mathbf{sat}_p^{\geqslant p_i}\ S_{P,i} \wedge Q \ \mathbf{sat}_p^{\geqslant q_i}\ S_{Q,i}$$

and

$$\forall i \in \{1, 2\} \quad S_{P,i}(\tau, \sqsubseteq_P, s) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s) \Rightarrow \text{silent}(\tau, \sqsubseteq_P \| \sqsubseteq_Q, s)$$

So we can use our rule to show that

$$P \| Q \ \mathbf{sat}_p^{\geqslant 0.54}\ \text{silent}$$

since $\sum_i p_i q_i = 0.54$, and $S_{P,1}$ and $S_{P,2}$ are disjoint so the side condition holds.

Larsen and Skou [LS92] have also investigated compositional verification of probabilistic processes, and they also find that they have to reduce a proof obligation on a composite process to a number of proof obligations on the subcomponents.

7.2.3 Combining multiple possibilities with conditional specifications

If having to deal with one of the above complications is not enough, we have to be able to deal with the case where both apply. The following proof rule covers both conditional specifications and multiple possibilities:

$$\frac{\begin{array}{l} \forall i : I \quad P \ \mathbf{sat}_p^{\geqslant p_i}\ S_{P,i} \mid G_{P,i} \\ \forall i : I \quad Q \ \mathbf{sat}_p^{\geqslant q_i}\ S_{Q,i} \mid G_{Q,i} \\ \forall i : I \quad \begin{pmatrix} s \subseteq A \cup B \wedge S_{P,i}(\tau, \sqsubseteq_P, s \quad A) \wedge G_{P,i}(\tau, \sqsubseteq_P, s \quad A) \\ \wedge\, S_{Q,i}(\tau, \sqsubseteq_Q, s \quad B) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s \quad B) \end{pmatrix} \Rightarrow \\ \qquad\qquad S(\tau, \sqsubseteq_P {}_A\|_B \sqsubseteq_Q, s) \wedge G(\tau, \sqsubseteq_P {}_A\|_B \sqsubseteq_Q, s) \\ \forall i : I \quad s \subseteq A \cup B \wedge G(\tau, \sqsubseteq_P {}_A\|_B \sqsubseteq_Q, s) \Rightarrow \\ \qquad\qquad G_{P,i}(\tau, \sqsubseteq_P, s \quad A) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s \quad B) \end{array}}{P \;{}_A\|_B\; Q \ \mathbf{sat}_p^{\geqslant \sum_i p_i q_i}\ S \mid G} \quad \left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ \textit{disjoint} \end{array}\right]$$

where

$$\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\widehat{=}} s \subseteq A \cup B \wedge S_{P,i}(\tau, \sqsubseteq_P, s \quad A) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s \quad B)$$

It is our sincere hope that we never have to use this rule, but that we can always make do with a simpler one

7.2.4 Universal quantification

Note that the specification $\forall i : I \quad P \;\mathbf{sat}_\rho^{\geqslant p}\; S_i$ is not the same as the specification $P \;\mathbf{sat}_\rho^{\geqslant p}\; \forall i : I \quad S_i$ — universal quantification does not commute with probabilistic specification. This is rather unfortunate as it means that we have to make a lot of our quantifications explicit when in a non-probabilistic setting we would normally make them implicit.

For example, consider a medium that loses a proportion of its inputs:

$$W \mathrel{\widehat{=}} \mu X \quad in \longrightarrow (out \longrightarrow X \;{}_p\sqcap_q\; X)$$

Let S_t (for $t \in TIME$) be defined by

$$S_t \mathrel{\widehat{=}} in \text{ at } t \Rightarrow out \text{ live from } t + \delta$$

S_t is the condition that if an in occurs at time t then it is correctly transmitted. It is certainly true that $\forall t : TIME \quad W \;\mathbf{sat}_\rho^{\geqslant p}\; S_t$: the probability of an input received at time t getting through is p. However, it is not the case that $W \;\mathbf{sat}_\rho^{\geqslant p}\; \forall t : TIME \quad S_t$. This latter specification says that the probability of *all* messages getting through is at least p.

It is interesting to note that this latter specification *is* satisfied by

$$W' \mathrel{\widehat{=}} (\mu X \quad in \longrightarrow out \longrightarrow X) \;{}_p\sqcap_q\; (\mu X \quad in \longrightarrow X)$$

The difference between the two specifications is related to the fact that recursion does not distribute through probabilistic choice.

We will sometimes want to prove that a composite process satisfies a number of related specifications; we can reduce this obligation to proving a number of specifications for the subcomponents, for example by using the following rule:

$$\frac{\begin{array}{l} \forall i : I \quad P \;\mathbf{sat}_\rho^{\geqslant p_i}\; S_{P,i} \\ \forall i : I \quad Q \;\mathbf{sat}_\rho^{\geqslant q_i}\; S_{Q,i} \\ \forall i : I \quad s \subseteq A \cup B \wedge S_{P,i}(\tau, \sqsubseteq_P, s \quad A) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s \quad B) \Rightarrow S_i(\tau, \sqsubseteq_P {}^{A}\|^{B} \sqsubseteq_Q, s) \end{array}}{\forall i : I \quad P \;{}^{A}\|^{B}\; Q \;\mathbf{sat}_\rho^{\geqslant p_i \, q_i}\; S_i}$$

When we have quantification of this form we will often make it implicit: we will pick an arbitrary $i \in I$ and prove that $P \;{}^{A}\|^{B}\; Q \;\mathbf{sat}_\rho^{\geqslant p_i \, q_i}\; S_i$ via the proof rule

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho^{\geqslant p_i}\; S_{P,i} \\ Q \;\mathbf{sat}_\rho^{\geqslant q_i}\; S_{Q,i} \\ s \subseteq A \cup B \wedge S_{P,i}(\tau, \sqsubseteq_P, s \quad A) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s \quad B) \Rightarrow S_i(\tau, \sqsubseteq_P {}^{A}\|^{B} \sqsubseteq_Q, s) \end{array}}{P \;{}^{A}\|^{B}\; Q \;\mathbf{sat}_\rho^{\geqslant p_i \, q_i}\; S_i}$$

For example, consider what happens when we chain two unreliable media together. For simplicity, assume the media are each only able to deal with one message: let

$$\begin{aligned} W_1 &\mathrel{\hat=} in \xrightarrow{1} (mid \longrightarrow STOP\ {}_p\sqcap_q\ STOP) \\ W_2 &\mathrel{\hat=} mid \xrightarrow{1} (out \longrightarrow STOP\ {}_{p'}\sqcap_{q'}\ STOP) \\ W &\mathrel{\hat=} W_1\ {}^{A}\|^{B}\ W_2 \end{aligned}$$

where $A \mathrel{\hat=} \{in, mid\}$, $B \mathrel{\hat=} \{mid, out\}$. We would like to show that if the environment always allows mid to occur, then out is offered within 2 seconds of an in with probability pp'. Pick $t \in TIME$; we will show

$$W\ \mathbf{sat}_\rho^{\geqslant pp'}\ S \qquad \text{where} \qquad S \mathrel{\hat=} \mathsf{internal}\ mid \wedge in\ \mathsf{at}\ t \Rightarrow out\ \mathsf{live\ from}\ t+2$$

Let

$$S_1 \mathrel{\hat=} in\ \mathsf{at}\ t \Rightarrow mid\ \mathsf{live\ from}\ t+1 \qquad S_2 \mathrel{\hat=} mid\ \mathsf{at}\ t+1 \Rightarrow out\ \mathsf{live\ from}\ t+2$$

It should be obvious that we can reduce the proof obligation to

$$W_1\ \mathbf{sat}_\rho^{\geqslant p}\ S_1 \qquad \text{and} \qquad W_2\ \mathbf{sat}_\rho^{\geqslant p'}\ S_2$$

since if $\Sigma s \subseteq A \cup B$ then $S_1(\tau, \sqsubseteq_1, s \upharpoonright A) \wedge S_2(\tau, \sqsubseteq_2, s \upharpoonright B) \Rightarrow S(\tau, \sqsubseteq_1 {}^{A}\|^{B} \sqsubseteq_2, s)$. Note that we have had to choose W_2's predicate very carefully so that the consequence of S_1 matches the antecedent of S_2.

As another example, suppose W_2 is as above but W_1 outputs after either one or two seconds:

$$W_1 \mathrel{\hat=} in \xrightarrow{1} (mid \longrightarrow STOP\ {}_p\sqcap_q\ WAIT\ 1\ ; mid \longrightarrow STOP)$$

We will show

$$W_1\ {}^{A}\|^{B}\ W_2\ \mathbf{sat}_\rho^{\geqslant p'}\ S$$

where

$$S \mathrel{\hat=} \mathsf{internal}\ mid \wedge in\ \mathsf{at}\ t \Rightarrow out\ \mathsf{live\ from}\ t+2 \vee out\ \mathsf{live\ from}\ t+3$$

where we are implicitly quantifying over t. Let

$$\begin{aligned} S_{1,1} &\mathrel{\hat=} in\ \mathsf{at}\ t \Rightarrow mid\ \mathsf{live\ from}\ t+1 \\ S_{1,2} &\mathrel{\hat=} in\ \mathsf{at}\ t \Rightarrow \mathsf{no}\ mid\ \mathsf{offered}\ [t+1, t+2) \wedge mid\ \mathsf{live\ from}\ t+2 \\ S_{2,1} &\mathrel{\hat=} mid\ \mathsf{at}\ t+1 \Rightarrow out\ \mathsf{live\ from}\ t+2 \\ S_{2,2} &\mathrel{\hat=} mid\ \mathsf{at}\ t+2 \Rightarrow out\ \mathsf{live\ from}\ t+3 \end{aligned}$$

It should be obvious that

$$W_1\ \mathbf{sat}_\rho^{\geqslant p}\ S_{1,1} \qquad W_1\ \mathbf{sat}_\rho^{\geqslant q}\ S_{1,2} \qquad W_2\ \mathbf{sat}_\rho^{\geqslant p'}\ S_{2,1} \qquad W_2\ \mathbf{sat}_\rho^{\geqslant p'}\ S_{2,2}$$

and

$$\forall i \in \{1,2\} \quad s \subseteq A \cup B \wedge S_{1,i}(\tau, \sqsubseteq_1, s \upharpoonright A) \wedge S_{2,i}(\tau, \sqsubseteq_2, s \upharpoonright B) \Rightarrow S(\tau, \sqsubseteq_1 {}^{A}\|^{B} \sqsubseteq_2, s)$$

so we can use the rule for parallel composition with multiple possibilities, as in section 7.2.2, to deduce that $W_1\ {}^{A}\|^{B}\ W_2\ \mathbf{sat}_\rho^{\geqslant p'}\ S$ since $pp' + qp' = p'$, and $S_{1,1}$ and $S_{1,2}$ are disjoint.

7.2.5 Simultaneous proof of several specifications

For recursion it is often not easy to prove that a process satisfies some probabilistic specification directly; it is more convenient to infer it from some more general result.

For example, consider the process $P \mathrel{\hat{=}} \mu X \quad a \xrightarrow{1} STOP\ {}_{1/2}\sqcap_{1/2}\ WAIT\ 1\ ; X$. We want to prove that this process offers an a within 3 seconds with a probability of at least 90%, i.e. $P\ \mathbf{sat}_\rho^{\geqslant 0.9}\ S$ where $S \mathrel{\hat{=}} \exists\, t \in [0,3] \quad a \text{ live } t$. We will prove this as a corollary of the following more general result:

$$\forall\, \imath : \qquad P\ \mathbf{sat}_\rho^{\geqslant p_\imath}\ S_\imath$$

where

$$S_\imath \mathrel{\hat{=}} \exists\, t \in [0,\imath] \quad a \text{ live } t \qquad p_i \mathrel{\hat{=}} 1 - (1/2)^{\imath+1}$$

S_i is the specification that an a is offered within $\imath$ seconds. Note that $S_3(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s)$ and $p_3 > 0.9$ so this will prove our requirement.

We have the following proof rule for recursion:

$$\frac{(\forall\, \imath \quad X\ \mathbf{sat}_\rho^{\geqslant p_\imath}\ S_i) \Rightarrow (\forall\, \imath \quad P\ \mathbf{sat}_\rho^{\geqslant p_\imath}\ S_i)}{\forall\, \imath \quad \mu X \quad P\ \mathbf{sat}_\rho^{\geqslant p_\imath}\ S_\imath}$$

Assume then that $\forall\, i \quad X\ \mathbf{sat}_\rho^{\geqslant p_\imath}\ S_\imath$ and pick $\imath \in$; we must show $P\ \mathbf{sat}_\rho^{\geqslant p_\imath}\ S_\imath$. We have the following rule for probabilistic choice:

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho^{\geqslant p'}\ S_P \\ Q\ \mathbf{sat}_\rho^{\geqslant q'}\ S_Q \\ S_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{P\ {}_p\sqcap_q\ Q\ \mathbf{sat}_\rho^{\geqslant p \cdot p' + q \cdot q'}\ S}$$

Let $S_P \mathrel{\hat{=}} S_0$, $p' \mathrel{\hat{=}} 1$, $S_Q \mathrel{\hat{=}} S_\imath$ and $q' \mathrel{\hat{=}} 1 - (1/2)^\imath$. Evidently $S_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \Rightarrow S_i(\tau, \sqsubseteq, s)$, and $p_\imath = 1/2.p' + 1/2.q'$ so we have reduced our proof obligation to

$$a \xrightarrow{1} STOP\ \mathbf{sat}_\rho^{\geqslant 1}\ S_0 \qquad \text{and} \qquad WAIT\ 1\ ; X\ \mathbf{sat}_\rho^{\geqslant 1-(1/2)^\imath}\ S_i$$

To prove the first of these, we can use rule 7.1.14 to reduce our proof obligation to $a \xrightarrow{1} STOP\ \mathbf{sat}_\rho\ S_0$, which can be easily proved using the proof rule for prefixing. When $\imath = 0$ the second proof obligation follows from rule 7.1.5; for $i > 0$, we use the proof rule for delay to reduce the proof obligation to $X\ \mathbf{sat}_\rho^{\geqslant 1-(1/2)^\imath}\ S_{\imath-1}$, which we have by the inductive hypothesis.

7.3 Derivation of the inference rules

In this section we derive proof rules for some of the constructs of our language. Rules for the rest of the constructs can be derived similarly. We handle the constructs in the following order:

- the basic processes $STOP$, $WAIT\ t$ and $SKIP$;
- the one-place operators of prefixing, delay, hiding and renaming;

- probabilistic choice;
- the two-place operators of external choice, parallel composition and interleaving;
- the transfer operators: sequential composition, timeout, timed transfer, and interrupt;
- recursion.

Most of the proofs were given in [Low92c]. Rules for all the operators are given in appendix B.3.

7.3.1 Basic processes

The basic processes $STOP$, $SKIP$ and $WAIT\ t$ are all completely deterministic, so the rules have the same form as in the unprobabilistic model. For example, we have the following rule for $STOP$.

$$\frac{S(\tau.[0,\tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec\succ)}{STOP\ \mathbf{sat}_\rho^{\geqslant 1}\ S}$$

7.3.2 One place operators

In this section we state a theory that can be used to derive a proof rule for the one place operators for prefixing, delay, hiding and renaming.

Theorem 7.3.1: Let F be a one place operator on the syntax of PBTCSP where if the environment condition b *compat* Ω is satisfied the semantic equation for F is of the form

$$\mathcal{P}_{PBT}\ F(P)\ \rho b = \begin{cases} 1 & \text{if } R(\Omega) \wedge b = f(\Omega) \\ 0 & \text{if } R(\Omega) \wedge b \neq f(\Omega) \\ \sum\{\!|\mathcal{P}_{PBT}\ \ P\ \rho b' \mid b = C(b',\Omega) \wedge T(b') \wedge b'\ compat\ \Omega'(\Omega)|\!\} & \text{if } \neg R(\Omega) \end{cases}$$

for some functions $R : EOFF \to Bool$, $f : EOFF \to BEH$ and $\Omega' : EOFF \to EOFF$ such that

$$\neg R(\Omega) \wedge b'\ compat\ \Omega'(\Omega) \Rightarrow T(b') \wedge C(b',\Omega)\ compat\ \Omega \qquad (*)$$

Then the following proof rule is sound:

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho^{\geqslant p}\ S_P(b) \mid G_P(b) \\ R(\Omega) \wedge b = f(\Omega) \wedge G(b) \Rightarrow S(b) \\ S_P(b) \wedge G_P(b) \wedge T(b) \Rightarrow S(C(b,\Omega)) \wedge G(C(b,\Omega)) \\ G(C(b)) \wedge T(b) \Rightarrow G_P(b) \end{array}}{F(P)\ \mathbf{sat}_\rho^{\geqslant p}\ S(b) \mid G(b)}$$

♡

In the case of some of the one place operators, the behaviours of $F(P)$ may not always depend upon the behaviours of P. For example, a behaviour of $a \xrightarrow{t} P$ ending at time τ will not depend upon any behaviour of P unless the environment offers an a no later than time $\tau - t$. We will define $R(\Omega)$ to be the predicate that is true precisely when the environment is such that the behaviour of $F(P)$ does not depend upon the behaviour of P. When $R(\Omega)$ holds, the behaviour of P will be a function of the environment, i.e. for some f we have $b = f(\Omega)$ with probability one. Premise 2 of the proof rule ensures that in this case $S(b)$ holds *whenever* $G(b)$ holds. For the hiding and renaming operators the behaviour of $F(P)$ will *always* depend upon the behaviour of P so we will take $R(\Omega) = \mathit{false}$.

If $R(\Omega)$ does not hold, then a behaviour of $F(P)$ will be a function of the corresponding behaviour of P and of the environment: $b = C(b', \Omega)$. Only certain behaviours of P are allowed: for example, if $F(P) = P \setminus X$, then P can only perform those behaviours where elements of X occur as soon as they are offered. The behaviour b' of P will be compatible with some environment Ω' which is a function of Ω; the condition $(*)$ relates Ω to Ω'. In this case, premise 4 ensures that G_P holds of P's behaviour whenever G holds of $F(P)$'s behaviour; premise 1 then ensures that with probability p, S_P holds of P's behaviour; in this case, premise 3 then ensures that S holds of $F(P)$'s behaviour.

This theorem was proved in [Low92c].

We can use the theorem to derive proof rules for the one place operators. For example, taking

$$\begin{aligned} R(\Omega) &\mathrel{\widehat=} \mathit{false} \\ C((\tau, \sqsubseteq, s), \Omega) &\mathrel{\widehat=} (\tau, \sqsubseteq \setminus X, s \setminus X) \\ T(\tau, \sqsubseteq, s) &\mathrel{\widehat=} s = \Uparrow_{\sqsubseteq}^{\setminus X}(s \setminus X) \\ \Omega'(\Omega) &\mathrel{\widehat=} \{v \mid v \setminus X \in \Omega\} \end{aligned}$$

we get the following rule for hiding:

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p}\ S_P \mid G_P \\ S_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{\setminus X} s) \wedge G_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{\setminus X} s) \Rightarrow S(\tau, \sqsubseteq \setminus X, s) \wedge G(\tau, \sqsubseteq \setminus X, s) \\ G(\tau, \sqsubseteq \setminus X, s) \Rightarrow G_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{\setminus X} s) \end{array}}{P \setminus X \ \mathbf{sat}_{\rho}^{\geqslant p}\ S \mid G}$$

Since $R(\Omega) = \mathit{false}$, the second premise in the general proof rule, above, disappears.

The above rule can be simplified to deal with unconditional specifications by taking $G(\tau, \sqsubseteq, s) = G_P(\tau, \sqsubseteq, s) = \mathit{true}$:

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p}\ S_P \\ S_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{\setminus X} s) \Rightarrow S(\tau, \sqsubseteq \setminus X, s) \end{array}}{P \setminus X \ \mathbf{sat}_{\rho}^{\geqslant p}\ S}$$

7.3.3 Probabilistic choice

We have the following proof rule for unconditional specifications.

$$\frac{\begin{array}{l} P \ \mathrm{sat}_{\rho}^{\geqslant p'} \ S_P \\ Q \ \mathrm{sat}_{\rho}^{\geqslant q'} \ S_Q \\ S_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{P \ {}_p\sqcap_q \ Q \ \mathbf{sat}_{\rho}^{\geqslant p \cdot p' + q \cdot q'} \ S}$$

The probability of $P\,{}_p\sqcap_q\,Q$ performing a behaviour that satisfies S is the probability of P being chosen (p) times the probability of P performing a behaviour that satisfies S (at least p') plus the probability of Q being chosen (q) times the probability of Q performing a behaviour that satisfies S (at least q').

For conditional specifications, the rule is slightly different.

$$\frac{\begin{array}{l} P \ \mathrm{sat}_{\rho}^{\geqslant p'} \ S_P \mid G_P \\ Q \ \mathrm{sat}_{\rho}^{\geqslant p'} \ S_Q \mid G_Q \\ S_P(\tau, \sqsubseteq, s) \wedge G_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \wedge G_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\ G(\tau, \sqsubseteq, s) \Rightarrow G_P(\tau, \sqsubseteq, s) \wedge G_Q(\tau, \sqsubseteq, s) \end{array}}{P \ {}_p\sqcap_q \ Q \ \mathbf{sat}_{\rho}^{\geqslant p'} \ S \mid G}$$

The reader may have been expecting a stronger rule than this, for example of the form

$$\frac{\begin{array}{l} P \ \mathrm{sat}_{\rho}^{\geqslant p'} \ S_P \mid G_P \\ Q \ \mathrm{sat}_{\rho}^{\geqslant q'} \ S_Q \mid G_Q \\ S_P(\tau, \sqsubseteq, s) \wedge G_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \wedge G_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\ G(\tau, \sqsubseteq, s) \Rightarrow G_P(\tau, \sqsubseteq, s) \wedge G_Q(\tau, \sqsubseteq, s) \end{array}}{P \ {}_p\sqcap_q \ Q \ \mathbf{sat}_{\rho}^{\geqslant pp' + qq'} \ S \mid G}$$

The reason we do not have a rule of this form is that given the premises and given that G holds of a behaviour of $P\ {}_p\sqcap_q\ Q$, we can say nothing about whether this is a behaviour of P or of Q. For example, let

$$P \mathrel{\hat=} (a \xrightarrow{1} (b \ {}_{p'}\sqcap_{1-p'} \ c)) \ {}_{p''}\sqcap_{1-p''} \ STOP \qquad Q \mathrel{\hat=} (a \xrightarrow{1} (b \ {}_{q'}\sqcap_{1-q'} \ c)) \ {}_{q''}\sqcap_{1-q''} \ STOP$$

and let

$$S_P = S_Q = S = b \ \mathsf{live} \ 1 \qquad G_P = G_Q = G = a \ \mathsf{at} \ 0$$

Then clearly all the premises of the above rule are satisfied, but for an environment Ω that offers an a at time 0

$$\frac{{}^{\Omega}_{P_p\sqcap_q Q} \ S \wedge G}{{}^{\Omega}_{P_p\sqcap_q Q} \ G} = \frac{pp'p'' + qq'q''}{pp'' + qq''}$$

which could be anything between p' and q' depending on the choice of p'' and q''.

7.3.4 Two place operators

In this section we state a theorem that can be used to derive proof rules for the external choice, parallel composition and interleaving operators.

Theorem 7.3.2: Let $\oplus$ be a binary operator on the syntax of PBTCSP that has a semantic equation with the following form:

$$\mathcal{P}_{PBT}\ P \oplus Q\ \rho b = \sum \{\!| \mathcal{P}_{PBT}\ P\ \rho b_P . \mathcal{P}_{PBT}\ Q\ \rho b_Q \mid b = b_P \oplus b_Q \wedge T(b_P, b_Q) |\!\}$$

where $\oplus$ is some binary operator on behaviours such that whenever $T(b_P, b_Q)$ holds we have

$$b_P\ compat\ \Omega_P(\Omega, b_P, b_Q) \wedge b_Q\ compat\ \Omega_Q(\Omega, b_P, b_Q) \Leftrightarrow T(b_P, b_Q) \wedge b_P \oplus b_Q\ compat\ \Omega \quad (*)$$

for some functions Ω_P and Ω_Q. Then the following proof rule is sound:

$$\frac{\begin{array}{ll} \forall i : I & P\ \mathbf{sat}_\rho^{\geqslant p_i}\ S_{P,i}(b) \mid G_{P,i}(b) \\ \forall i : I & Q\ \mathbf{sat}_\rho^{\geqslant q_i}\ S_{Q,i}(b) \mid G_{Q,i}(b) \\ \forall i : I & S_{P,i}(b_P) \wedge G_{P,i}(b_P) \wedge S_{Q,i}(b_Q) \wedge G_{Q,i}(b_Q) \wedge T(b_P, b_Q) \Rightarrow \\ & \quad S(b_P \oplus b_Q) \wedge G(b_P \oplus b_Q) \\ \forall i : I & G(b_P \oplus b_Q) \wedge T(b_P, b_Q) \Rightarrow G_{P,i}(b_P) \wedge G_{Q,i}(b_Q) \end{array}}{P \oplus Q\ \mathbf{sat}_\rho^{\geqslant \sum_i p_i q_i}\ S(b) \mid G(b)} \left[\begin{array}{c} \langle \hat{S}_i(b_P, b_Q) \mid i : I \rangle \\ disjoint \end{array}\right]$$

where $\hat{S}_i(b_P, b_Q) \mathrel{\hat{=}} T(b_P, b_Q) \wedge S_{P,i}(b_P) \wedge G_{P,i}(b_P) \wedge S_{Q,i}(b_Q) \wedge G_{Q,i}(b_Q)$. ♡

If a behaviour of $P \oplus Q$ satisfies G then premise 4 ensures that the corresponding behaviours of P and Q satisfy $G_{P,i}$ and $G_{Q,i}$, for all i. P and Q are evaluated in environments Ω_P and Ω_Q; $(*)$ relates these to the environment for $P \oplus Q$. Premise 1 then ensures that the behaviour of P satisfies $S_{P,i}$ with probability p_i, and premise 2 ensures that the behaviour of Q satisfies $S_{Q,i}$ with probability q_i. Premise 3 then tells us that the behaviour of $P \oplus Q$ satisfies S. Because of the side condition, it is valid to sum over all i, so we see that the behaviour of $P \oplus Q$ satisfies S with probability at least $\sum_i p_i q_i$.

The proof appeared in [Low92c].

We can use this theorem to derive rules for the two-place operators. For example, taking

$$\begin{array}{rcl} (\tau_P, \sqsubseteq_P, s_P) \oplus (\tau_Q, \sqsubseteq_Q, s_Q) & \hat{=} & (\tau_P, \sqsubseteq_P \mathbin{+\!\!\!+} \sqsubseteq_Q, s) \\ T((\tau_P, \sqsubseteq_P, s_P), (\tau_Q, \sqsubseteq_Q, s_Q)) & \hat{=} & \tau_P = \tau_Q \wedge s_P = s_Q \\ \Omega_P(\Omega, (\tau_P, \sqsubseteq_P, s_P), (\tau_Q, \sqsubseteq_Q, s_Q)) & \hat{=} & \Omega \cap \mathsf{items} \sqsubseteq_Q \\ \Omega_Q(\Omega, (\tau_P, \sqsubseteq_P, s_P), (\tau_Q, \sqsubseteq_Q, s_Q)) & \hat{=} & \{(t, \sqcup_{\sqsubseteq_P}(\Omega \uparrow t \cap \mathsf{items} \sqsubseteq_Q)) \mid t \quad \tau_P\} \end{array}$$

we have the following proof rule for parallel composition

$$\frac{\begin{array}{ll} \forall i & P\ \mathbf{sat}_\rho^{\geqslant p_i}\ S_{P,i} \mid G_{P,i} \\ \forall i & Q\ \mathbf{sat}_\rho^{\geqslant q_i}\ S_{Q,i} \mid G_{Q,i} \\ \forall i & \begin{pmatrix} S_{P,i}(\tau, \sqsubseteq_P, s) \wedge G_{P,i}(\tau, \sqsubseteq_P, s) \\ \wedge\ S_{Q,i}(\tau, \sqsubseteq_Q, s) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s) \end{pmatrix} \Rightarrow \begin{pmatrix} S(\tau, \sqsubseteq_P \mathbin{+\!\!\!+} \sqsubseteq_Q, s) \\ \wedge\ G(\tau, \sqsubseteq_P \mathbin{+\!\!\!+} \sqsubseteq_Q, s) \end{pmatrix} \\ \forall i & G(\tau, \sqsubseteq_P \mathbin{+\!\!\!+} \sqsubseteq_Q, s) \Rightarrow G_{P,i}(\tau, \sqsubseteq_P, s) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s) \end{array}}{P \mathbin{+\!\!\!+} Q\ \mathbf{sat}_\rho^{\geqslant \sum_i p_i q_i}\ S \mid G} \left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ disjoint \end{array}\right]$$

where $\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\hat{=}} S_{P,i}(\tau, \sqsubseteq_P, s) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s)$.

Note that there are simpler forms for this rule where we consider unconditional specifications, or we reduce the proof obligation to a single proof obligation on the subcomponents.

- where we consider unconditional specifications:

$$\frac{\begin{array}{l} \forall\, i \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_{P,i} \\ \forall\, i \quad Q\ \mathbf{sat}_{\rho}^{\geqslant q_i}\ S_{Q,i} \\ \forall\, i \quad S_{P,i}(\tau, \sqsubseteq_P, s) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \mathbin{+\!\!\!\!+\!\!\!\!\!\!\;|\,|} \sqsubseteq_Q, s) \end{array}}{P \mathbin{|||} Q\ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i p_i q_i}\ S} \left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ \textit{disjoint} \end{array}\right]$$

where $\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\widehat{=}} S_{P,i}(\tau, \sqsubseteq_P, s) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s)$.

- where we reduce the proof obligation to a single proof obligation on the subcomponents:

$$\frac{\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S_P \mid G_P \\ Q\ \mathbf{sat}_{\rho}^{\geqslant q}\ S_Q \mid G_Q \\ \begin{pmatrix} S_P(\tau, \sqsubseteq_P, s) \wedge G_P(\tau, \sqsubseteq_P, s) \\ \wedge\ S_Q(\tau, \sqsubseteq_Q, s) \wedge G_Q(\tau, \sqsubseteq_Q, s) \end{pmatrix} \Rightarrow S(\tau, \sqsubseteq_P \mathbin{|||} \sqsubseteq_Q, s) \wedge G(\tau, \sqsubseteq_P \mathbin{|||} \sqsubseteq_Q, s) \\ G(\tau, \sqsubseteq_P \mathbin{|||} \sqsubseteq_Q, s) \Rightarrow G_P(\tau, \sqsubseteq_P, s) \wedge G_Q(\tau, \sqsubseteq_Q, s) \end{array}}{P \mathbin{|||} Q\ \mathbf{sat}_{\rho}^{\geqslant pq}\ S \mid G}$$

- where we make both simplifications:

$$\frac{\begin{array}{l} P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S_P \\ Q\ \mathbf{sat}_{\rho}^{\geqslant q}\ S_Q \\ S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \mathbin{|||} \sqsubseteq_Q, s) \end{array}}{P \mathbin{|||} Q\ \mathbf{sat}_{\rho}^{\geqslant pq}\ S}$$

7.3.5 Transfer operators

In this section we state a theorem that can be used to derive rules for the sequential composition, timeout and timed transfer operators. If we write $\rightsquigarrow$ for one of these operators, then $P \rightsquigarrow Q$ initially acts "like" P (strictly the behaviour of $P \rightsquigarrow Q$ is derived from a behaviour of P); then, according to certain circumstances, control is transferred to Q. Writing f_P for $\mathcal{P}_{PBT}\ P\ \rho$ and f_Q for $\mathcal{P}_{PBT}\ Q\ \rho$, the probability function for each of these operators can be written as

$$\begin{array}{l} \mathcal{P}_{PBT}\ P \rightsquigarrow Q\ \ \rho(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} \\ \quad \sum \left\{\!\!\left| f_P(\tau_P, \sqsubseteq_P, s_P) \;\middle|\; \begin{array}{l} ok(\tau_P, \sqsubseteq_P, s_P) \wedge \text{no transfer}(\tau_P, \sqsubseteq_P, s_P) \\ \wedge\ (\tau, \sqsubseteq, s) = C(\tau_P, \sqsubseteq_P, s_P) \end{array} \right|\!\!\right\} \\ \quad + \sum \left\{\!\!\left| f_P(\tau_P, \sqsubseteq_P, s_P) \;\middle|\; \begin{array}{l} ok(\tau_P, \sqsubseteq_P, s_P) \wedge (\tau_P, \sqsubseteq_P, s_P) \text{ transfer at } t \\ \wedge\ t \quad \tau < t + \delta \wedge (\tau, \sqsubseteq, s) = C(\tau_P, \sqsubseteq_P, s_P) \quad \text{empty}_{(t,\tau]} \end{array} \right|\!\!\right\} \\ \quad + \sum \left\{\!\!\left| \begin{array}{l} f_P(\tau_P, \sqsubseteq_P, s_P).f_Q(\tau_Q, \sqsubseteq_Q, s_Q) \mid \\ \qquad ok(\tau_P, \sqsubseteq_P, s_P) \wedge (\tau_P, \sqsubseteq_P, s_P) \text{ transfer at } t \wedge t + \delta \quad \tau \\ \qquad \wedge\ (\tau, \sqsubseteq, s) = C(\tau_P, \sqsubseteq_P, s_P) \quad \text{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta \end{array} \right|\!\!\right\} \end{array}$$

The predicate $(\tau, \sqsubseteq, s)$ transfer at t is true if control should be removed from P at time t; for sequential composition it is the condition that $t = \tau$ and a occurs at t. The predicate no transfer$(\tau, \sqsubseteq, s)$ is equivalent to $\forall\, t \quad \tau \quad \neg\ (\tau, \sqsubseteq, s) \quad t$ transfer at t: it is true if control should remain with P throughout the behaviour $(\tau, \sqsubseteq, s)$; for sequential composition it is the condition that no occurs. The function C changes a behaviour of P into one of $P \leadsto Q$; for sequential composition it hides all s. The predicate $ok(b_P)$ is true if b_P is a behaviour that P could perform while in the combination $P \leadsto Q$; for sequential composition it is the condition that a is never refused. empty_I is the empty behaviour during time interval I: for example, $(t, \sqsubseteq, s)\ \ \mathrm{empty}_{(t,\tau]} = (\tau, \sqsubseteq\ \ (t,\tau] \otimes \langle\!\{\}\!\rangle, s\ \ \prec\succ)$.

For each of these operators there is a function $\Omega' : EOFF \to EOFF$ such that whenever $ok(b_P)$ holds,

$$
\begin{aligned}
C(b_P)\ compat\ \Omega &\Leftrightarrow b_P\ compat\ \Omega'(\Omega) \\
C(b_P)\ \ \mathrm{empty}_{(t,\tau]}\ compat\ \Omega &\Leftrightarrow b_P\ compat\ \Omega'(\Omega)\ \ t \\
C(b_P)\ \ \mathrm{empty}_{(t,t+\delta)}\ \ b_Q + t + \delta\ compat\ \Omega &\Leftrightarrow b_P\ compat\ \Omega'(\Omega)\ \ t \wedge b_Q\ compat\ \Omega - t - \delta
\end{aligned}
$$

Informally, $\Omega'(\Omega)$ is the environment that P encounters up until the time of transfer when $P \leadsto Q$ is in environment Ω. For sequential composition it is the environment that offers whatever Ω offers along with as many s as P can perform.

We have the following proof rule:

Rule 7.3.3 (Transfer operators)

$$
\frac{
\begin{array}{ll}
\forall i & P\ \mathbf{sat}_\rho^{\geqslant p_i}\ S_{P,i}(b) \mid G_{P,i}(b) \wedge ok(b) \wedge \text{no transfer } b \\
\forall i & P\ \mathbf{sat}_\rho^{\geqslant p_i'}\ S'_{P,i}(b) \mid G'_{P,i}(b) \wedge ok(b) \wedge b \text{ transfer at } t \\
\forall i & Q\ \mathbf{sat}_\rho^{\geqslant q_i}\ S_{Q,i}(b) \mid G_{Q,i}(b) \\
\forall i & S_{P,i}(b) \wedge G_{P,i}(b) \wedge ok(b) \wedge \text{no transfer } b \Rightarrow S(C(b)) \wedge G(C(b)) \\
\forall i & \begin{pmatrix} S'_{P,i}(b) \wedge G'_{P,i}(b) \wedge ok(b) \\ \wedge\ b \text{ transfer at } t \wedge t\ \ \tau < t + \delta \end{pmatrix} \Rightarrow \begin{pmatrix} S(C(b)\ \ \mathrm{empty}_{(t,\tau]}) \\ \wedge\ G(C(b)\ \ \mathrm{empty}_{(t,\tau]}) \end{pmatrix} \\
\forall i & \begin{pmatrix} S'_{P,i}(b_P) \wedge G'_{P,i}(b_P) \wedge S_{Q,i}(b_Q) \\ \wedge\ G_{Q,i}(b_Q) \wedge ok(b) \wedge b_P \text{ transfer at } t \end{pmatrix} \Rightarrow \\
& \quad \begin{pmatrix} S(C(b_P)\ \ \mathrm{empty}_{(t,t+\delta)}\ \ b_Q + t + \delta) \\ \wedge\ G(C(b_P)\ \ \mathrm{empty}_{(t,t+\delta)}\ \ b_Q + t + \delta) \end{pmatrix} \\
\forall i & G(C(b)) \wedge ok(b) \wedge \text{no transfer } b \Rightarrow G_{P,i}(b) \\
\forall i & \begin{pmatrix} G(C(b)\ \ \mathrm{empty}_{(t,\tau]}) \wedge ok(b) \\ \wedge\ b \text{ transfer at } t \wedge t\ \ \tau < t + \delta \end{pmatrix} \Rightarrow G'_{P,i}(b) \\
\forall i & \begin{pmatrix} G(C(b)\ \ \mathrm{empty}_{(t,t+\delta)}\ \ b_Q + t + \delta) \\ \wedge\ ok(b) \wedge b \text{ transfer at } t \end{pmatrix} \Rightarrow G'_{P,i}(b) \wedge G_{Q,i}(b_Q)
\end{array}
}{
P \leadsto Q\ \mathbf{sat}_\rho^{\geqslant \Sigma_i p_i}\ S(b) \mid G(b)
}
\left[\begin{array}{l} \langle \hat{S}_i(b) \rangle\ disjoint \\ \langle \hat{S}'_i(b) \rangle\ disjoint \\ \Sigma_i\, p'_i q_i\ \ \Sigma_i\, p_i \end{array}\right]
$$

where i ranges over some set I and

$$
\begin{aligned}
\hat{S}_i(b) &\cong ok(b) \wedge \text{no transfer } b \wedge S_{P,i}(b) \wedge G_{P,i}(b) \\
\hat{S}'_i(b) &\cong ok(b) \wedge b \text{ transfer at } t \wedge S'_{P,i}(b) \wedge G'_{P,i}(b)
\end{aligned}
$$

$\triangle$

This rule was proved sound in [Low92c]. The first three premises give predicates satisfied by P and Q: note that we give different predicates for P in the cases where transfer does or does not happen; in many applications we will take these predicates to be the same. The next three premises say that if behaviours of P and Q satisfy their predicates then the resulting behaviour of $P \rightsquigarrow Q$ satisfies its predicates. The last three premises say that if a behaviour of $P \rightsquigarrow Q$ satisfies G then the corresponding behaviours of P and Q satisfy their "given" predicates (i.e. the predicates appearing on the right hand of the '|' in the first three premises).

We will now use this rule to derive a rule for the timed transfer operator, $P \;_t\; Q$. We take

$$\begin{aligned} (\tau, \sqsubseteq, s) \text{ transfer at } t' &\Leftrightarrow t = t' = \tau \\ \text{no transfer}(\tau, \sqsubseteq, s) &\Leftrightarrow \tau < t \\ ok(\tau, \sqsubseteq, s) &\Leftrightarrow \mathit{true} \\ C(\tau, \sqsubseteq, s) &\mathrel{\widehat{=}} (\tau, \sqsubseteq, s) \\ \Omega'(\Omega) &\mathrel{\widehat{=}} \Omega \end{aligned}$$

This gives us the following rule:

$$\begin{array}{l} \forall i \;\; P\,\mathbf{sat}_\rho^{\geqslant p_i}\; S_{P,i}(\tau, \sqsubseteq, s) \mid G_{P,i}(\tau, \sqsubseteq, s) \wedge \tau < t \\ \forall i \;\; P\,\mathbf{sat}_\rho^{\geqslant p'_i}\; S'_{P,i}(\tau, \sqsubseteq, s) \mid G'_{P,i}(\tau, \sqsubseteq, s) \wedge \tau = t \\ \forall i \;\; Q\,\mathbf{sat}_\rho^{\geqslant q_i}\; S_{Q,i}(\tau, \sqsubseteq, s) \mid G_{Q,i}(\tau, \sqsubseteq, s) \\ \forall i \;\; S_{P,i}(\tau, \sqsubseteq, s) \wedge G_{P,i}(\tau, \sqsubseteq, s) \wedge \tau < t \Rightarrow S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\ \forall i \;\; S'_{P,i}(t, \sqsubseteq, s) \wedge G'_{P,i}(t, \sqsubseteq, s) \wedge t \quad \tau < t + \delta \Rightarrow \\ \qquad S((t, \sqsubseteq, s) \quad \mathrm{empty}_{(t,\tau]}) \wedge G((t, \sqsubseteq, s) \quad \mathrm{empty}_{(t,\tau]}) \\ \forall i \;\; \begin{pmatrix} S'_{P,i}(t, \sqsubseteq_P, s_P) \wedge G'_{P,i}(t, \sqsubseteq_P, s_P) \\ \wedge\, S_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \wedge G_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \end{pmatrix} \Rightarrow \\ \qquad \begin{pmatrix} S((t, \sqsubseteq_P, s_P) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \\ \wedge\, G((t, \sqsubseteq_P, s_P) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \end{pmatrix} \\ \forall i \;\; G(\tau, \sqsubseteq, s) \wedge \tau < t \Rightarrow G_{P,i}(\tau, \sqsubseteq, s) \\ \forall i \;\; G((t, \sqsubseteq, s) \quad \mathrm{empty}_{(t,\tau]}) \wedge t \quad \tau < t + \delta \Rightarrow G'_{P,i}(t, \sqsubseteq, s) \\ \forall i \;\; G((t, \sqsubseteq_P, s_P) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \Rightarrow \\ \qquad G'_{P,i}(t, \sqsubseteq_P, s_P) \wedge G_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \\ \hline P \;_t\; Q\,\mathbf{sat}_\rho^{\geqslant \Sigma_i p_i}\; S \mid G \end{array} \quad \begin{bmatrix} \langle \hat{S}_i(\tau, \sqsubseteq, s) \rangle \\ \qquad \mathit{disjoint} \\ \langle \hat{S}'_i(\tau, \sqsubseteq, s) \rangle \\ \qquad \mathit{disjoint} \\ \Sigma_i\, p'_i q_i \quad \Sigma_i\, p_i \end{bmatrix}$$

where i ranges over some set I and

$$\begin{aligned} \hat{S}_i(\tau, \sqsubseteq, s) &\mathrel{\widehat{=}} \tau < t \wedge S_{P,i}(\tau, \sqsubseteq, s) \wedge G_{P,i}(\tau, \sqsubseteq, s) \\ \hat{S}'_i(\tau, \sqsubseteq, s) &\mathrel{\widehat{=}} \tau = t \wedge S'_{P,i}(\tau, \sqsubseteq, s) \wedge G'_{P,i}(\tau, \sqsubseteq, s) \end{aligned}$$

The above rule can be simplified in the normal ways by considering unconditional specifications, or by considering only single specifications for the components.

Interrupts

By analogy with the previous operators, one might expect to have a proof rule for the interrupt operator of the following form, ignoring conditional specifications and considering only a single specification for each component:

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p}\ S_P \\ Q \ \mathbf{sat}_{\rho}^{\geqslant q}\ S_Q \\ S_P(\tau, \sqsubseteq, s) \wedge e \notin \Sigma s \Rightarrow S(\tau, \sqsubseteq \oplus e, s) \\ S_P(t, \sqsubseteq, s) \wedge e \notin \Sigma s \wedge t \quad \tau < t + \delta \Rightarrow S((t, \sqsubseteq \oplus e, s \quad (t, e)) \quad \mathsf{empty}_{(t,\tau]}) \\ S_P(t, \sqsubseteq, s) \wedge e \notin \Sigma s \wedge S_Q(b_Q) \Rightarrow S((t, \sqsubseteq \oplus e, s \quad (t, e)) \quad \mathsf{empty}_{(t,t+\delta)} \quad b_Q + t + \delta) \end{array}}{P \underset{e}{\nabla} Q \ \mathbf{sat}_{\rho}^{\geqslant pq}\ S}$$

However, this is not the case. Consider the processes

$$P \mathrel{\widehat{=}} (a \xrightarrow{1} SKIP \ {}_{1/2}\sqcap_{1/2}\ WAIT\ 1) \, ; \mu X \quad a \xrightarrow{1} X \qquad Q \mathrel{\widehat{=}} STOP$$

It is easily seen that in an environment that is always willing to perform as, the probability that P performs an even number of as is $1/2$:

$$P \ \mathbf{sat}_{\rho}^{\geqslant 1/2}\ S_P \qquad \text{where} \qquad S_P \mathrel{\widehat{=}} \mathsf{internal}\ a \Rightarrow \mathsf{count}\ a\ \mathsf{even}$$

and Q performs no as:

$$Q \ \mathbf{sat}_{\rho}^{\geqslant 1}\ S_Q \qquad \text{where} \qquad S_Q \mathrel{\widehat{=}} \mathsf{count}\ a = 0$$

If we define S by

$$S \mathrel{\widehat{=}} \mathsf{internal}\ a \Rightarrow \mathsf{count}\ a\ \mathsf{even}$$

then it is easy to see that the third, fourth and fifth premises of the "proof rule" are satisfied. However, we do not have the consequent, for consider an environment that always offers the interrupt event after one a has been performed — an environment that can be achieved by placing this process in parallel with $a \xrightarrow{1/2} e \longrightarrow STOP$, for example: in this case the process always performs precisely one a, so S is never satisfied. In a sense this is because the interrupt mechanism conspires against the predicate S, forcing the interrupt at a time that makes S false.

However, the above rule is correct if we restrict S_P to being a safety predicate: a predicate such that if it is true of a behaviour is also true of any prefix of that behaviour. We formally define safety predicates by:

Definition 7.3.4: A predicate S is a safety predicate if whenever $S(\tau, \sqsubseteq, s)$ holds and $t \quad \tau$ then we have $S(t, \sqsubseteq \quad t, s \quad t)$. $\diamond$

We have the following rule, which was proved in [Low92c].

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p}\ S_P \\ Q \ \mathbf{sat}_{\rho}^{\geqslant q}\ S_Q \\ S_P(\tau, \sqsubseteq, s) \wedge e \notin \Sigma s \Rightarrow S(\tau, \sqsubseteq \oplus e, s) \\ S_P(t, \sqsubseteq, s) \wedge e \notin \Sigma s \wedge t \quad \tau < t + \delta \Rightarrow \\ \qquad S((t, \sqsubseteq \oplus e, s \quad (t, e)) \quad \mathsf{empty}_{(t,\tau]}) \\ S_P(t, \sqsubseteq, s) \wedge e \notin \Sigma s \wedge S_Q(b_Q) \Rightarrow \\ \qquad S((t, \sqsubseteq \oplus e, s \quad (t, e)) \quad \mathsf{empty}_{(t,t+\delta)} \quad b_Q + t + \delta) \end{array}}{P \underset{e}{\nabla} Q \ \mathbf{sat}_{\rho}^{\geqslant pq}\ S} \left[S_P \text{ is a safety predicate} \right]$$

7.3.6 Recursion

In this section we derive proof rules for immediate recursion. Rules for delayed recursion and mutual recursion can be derived similarly.

If P is constructive for X then we have the following proof rule for immediate recursion:

Rule 7.3.5:

$$\frac{\forall\, Y : \mathcal{PP}_{TB} \quad R(Y) \Rightarrow R(\mathcal{F}_{PBT} \;\; P \;\; \rho[Y/X])}{R(\mathcal{F}_{PBT} \;\; \mu\, X \quad P \;\; \rho)} \left[R \text{ continuous and satisfiable} \right]$$

△

Proof: The proof of this is identical to the proof of rule 5.4.3. □

We have the following proof rule for probabilistic specifications:

Rule 7.3.6 (Recursion)

$$\frac{(\forall\, i \quad X \;\mathbf{sat}_{\rho}^{\geqslant p_i}\; S_i \mid G_i) \;\Rightarrow\; (\forall\, i \quad P \;\mathbf{sat}_{\rho}^{\geqslant p_i}\; S_i \mid G_i)}{\forall\, i \quad \mu\, X \quad P \;\mathbf{sat}_{\rho}^{\geqslant p_i}\; S_i \mid G_i}$$

for P constructive for X, where i ranges over some set I. △

We need the following result adapted from [Ree88]:

Theorem 7.3.7: A specificatidn R is continuous if for all X in $\mathcal{PP}_{TB}$ such that $R(X) = \mathit{false}$:

$$\exists\, t : \mathit{TIME} \quad \forall\, Y : \mathcal{PP}_{TB} \quad Y \quad t = X \quad t \Rightarrow R(Y) = \mathit{false}$$

♡

We can now prove the inference rule sound:

Proof of rule 7.3.6: In order to use rule 7.3.5 we only need to prove that the predicate

$$R(Y) \mathrel{\widehat{=}} \forall\, i \quad \forall\, \Omega \qquad \overset{\Omega}{\underset{Y}{}} \;\; S_i \wedge G_i \qquad p_i \;\; \overset{\Omega}{\underset{Y}{}} \;\; G_i$$

is continuous and satisfiable, where the notation is extended to arbitrary members of $\mathcal{PP}_{TB}$ in the obvious way:

$$\overset{\Omega}{\underset{X}{}} \;\; S \;\mathrel{\widehat{=}}\; \sum\left\{\!\!\left| \pi_2 X(\tau, \sqsubseteq, s) \mid (\tau, \sqsubseteq, s) \;\mathit{compat}\; \Omega \wedge S(\tau, \sqsubseteq, s) \right|\!\!\right\}$$

To show continuity, suppose that $X \in \mathcal{PP}_{TB}$ and $R(X) = \mathit{false}$. Then for some $\Omega \in \mathit{EOFF}$ and $i \in I$

$$\overset{\Omega}{\underset{X}{}} \;\; S_i \wedge G_i \;\; < p_i \;\; \overset{\Omega}{\underset{X}{}} \;\; G_i$$

Let $t \mathrel{\hat{=}} end\,\Omega$ and pick $Y \in \mathcal{PP}_{TB}$ such that $Y \quad t = X \quad t$. In order to apply theorem 7.3.7 we must show $R(Y) = false$. For any behaviour $(\tau, \sqsubseteq, s)$, if $(\tau, \sqsubseteq, s)$ *compat* Ω then $\tau = t$ and so $\pi_2 Y(\tau, \sqsubseteq, s) = \pi_2 X(\tau, \sqsubseteq, s)$. Hence for any predicate T,

$$\begin{aligned} {}^{\Omega}_{Y} \quad T &= \sum\{\!\!\{\pi_2 Y(\tau, \sqsubseteq, s) \mid (\tau, \sqsubseteq, s) \; compat\; \Omega \wedge T(\tau, \sqsubseteq, s)\}\!\!\} \\ &= \sum\{\!\!\{\pi_2 X(\tau, \sqsubseteq, s) \mid (\tau, \sqsubseteq, s) \; compat\; \Omega \wedge T(\tau, \sqsubseteq, s)\}\!\!\} \\ &= {}^{\Omega}_{X} \quad T \end{aligned}$$

so in particular

$$\begin{aligned} {}^{\Omega}_{Y} \quad S_i \wedge G_i &= {}^{\Omega}_{X} \quad S_i \wedge G_i \\ &< p_i \cdot {}^{\Omega}_{X} \quad G_i \\ &= p_i \cdot {}^{\Omega}_{Y} \quad G_i \end{aligned}$$

so $R(Y) = false$, as required.

For satisfiability, consider $X \mathrel{\hat{=}} (\{\}, \lambda(\tau, \sqsubseteq, s) \quad 0)$. Then for all $i \in I$ and $\Omega \in EOFF$ we have ${}^{\Omega}_{X} \quad G_i \; = 0$, so ${}^{\Omega}_{X} \quad S_i \wedge G_i \quad p_i \cdot {}^{\Omega}_{X} \quad G_i$. Hence $R(X)$ holds. □

Note that in proving the premise of the proof rule we cannot assume that X is a member of $\mathcal{PP}_{TB}$: we may not assume that any of the axioms are satisfied by X. This is rarely a problem.

For unconditional specifications we have the following proof rule:

Rule 7.3.8:

$$\frac{(\forall i \quad X \,\mathbf{sat}_\rho^{\geqslant p_i}\, S_i) \Rightarrow (\forall i \quad P \,\mathbf{sat}_\rho^{\geqslant p_i}\, S_i)}{\forall i \quad \mu X \quad P \,\mathbf{sat}_\rho^{\geqslant p_i}\, S_i} \left[\forall i \quad \forall \Omega \quad \exists\, b_\Omega \quad b_\Omega \; compat\; \Omega \wedge S_i(b_\Omega)\right]$$

△

where P is constructive for X. The side condition says that for each environment we can find a behaviour that satisfies S_i.

Proof: As in the previous case, it is enough to show that the predicate

$$R(Y) \mathrel{\hat{=}} \forall i \quad \forall \Omega \quad {}^{\Omega}_{Y} \quad S_i \qquad p_i$$

is continuous and satisfiable. The proof of continuity is as before. For satisfiability, consider $X \mathrel{\hat{=}} (\{\}, \lambda(\tau, \sqsubseteq, s) \quad 1)$. For all environments Ω and for all i, we have, by the side condition, ${}^{\Omega}_{X} \quad S_i \quad \pi_2 X(b_\Omega) = 1 \quad p_i$. Hence $R(X)$ holds as required. □

The form of the side condition is not very convenient. The following rule has a side condition that is generally easier to prove.

Rule 7.3.9 (Delayed recursion)

$$\frac{(\forall i \quad X \,\mathbf{sat}_\rho^{\geqslant p_i}\, S_i) \Rightarrow (\forall i \quad P \,\mathbf{sat}_\rho^{\geqslant p_i}\, S_i)}{\forall i \quad \mu X \quad P \,\mathbf{sat}_\rho^{\geqslant p_i}\, S_i} \left[\forall i \quad \exists P : PBTCSP \quad P \,\mathbf{sat}_\rho\, S_i\right]$$

△

It is enough to find, for each i, a process that satisfies S_i.

Proof: We will show that the side condition of this rule implies the side condition of the previous rule. Pick i; then there exists a process P such that $P\ \mathbf{sat}_\rho\ S_i$. Now for all environments Ω,

$$\sum \{\!\!| \mathcal{P}_{PBT}\ P\ \rho(\tau, \sqsubseteq, s) \mid (\tau, \sqsubseteq, s)\ compat\ \Omega |\!\!\} = 1$$

by theorem 4.2.2. In particular, there is some behaviour b_Ω such that b_Ω *compat* Ω and $\mathcal{P}_{PBT}\ P\ \rho(b_\Omega) > 0$. Then $b_\Omega \in \mathcal{A}_{PBT}\ P\ \rho$ by axiom P4 of the semantic space, and so $S_i(b_\Omega)$ since $P\ \mathbf{sat}_\rho\ S_i$. □

Note that all the above rules can be simplified by taking I to be a singleton set.

7.4 Case study: a simple protocol

In this section we consider a very simple communications protocol, illustrated in figure 7.2. Messages are input on the channel *in*. S then tries transmitting them over the medium M, which loses a proportion of its inputs. If the message is received by R then it is acknowledged and then output. If S does not receive an acknowledgement within a certain time then it times-out and re-transmits.

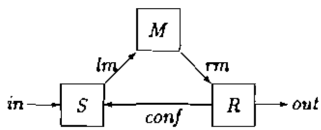

Figure 7.2: A simple protocol

We shall write *in* for $\{in.x \mid x \in X\}$, etc., where X is the type of the data transmitted. We shall also feel free to abuse notation by writing, for example, lm, rm for $lm \cup rm$.

The definitions of the processes are

$$\begin{aligned}
PROT &\mathrel{\hat=} PROT' \setminus A \\
PROT' &\mathrel{\hat=} (S \underset{conf}{\parallel} R) \underset{lm,rm}{\parallel} M \\
S &\mathrel{\hat=} in?x \longrightarrow lm!x \longrightarrow S'(x) \\
S'(x) &\mathrel{\hat=} conf_1 \longrightarrow conf_2 \longrightarrow S \ {}^{t_0}\ lm!x \longrightarrow S'(x) \\
M &\mathrel{\hat=} lm?x \xrightarrow{t_0} (rm!x \longrightarrow M\ {}_p\sqcap_q\ WAIT\ \delta\ ; M) \\
R &\mathrel{\hat=} rm?x \longrightarrow conf_1 \longrightarrow out!x \longrightarrow conf_2 \longrightarrow R
\end{aligned}$$

where $A \mathrel{\hat=} \{lm, rm, conf\}$. The length of the time out is chosen to ensure that S does not time out before the acknowledgement can get through. For convenience we name the alphabets of S and R:

$$A_S \mathrel{\hat=} \{in, lm, conf\} \qquad A_R \mathrel{\hat=} \{rm, out, conf\}$$

In section 7.4.1 we will prove that the protocol acts like a one place buffer, i.e. the output stream of data is a prefix of the input stream of data and is at most one item shorter. This proof will not require any treatment of probabilities: it can be carried out using only the proof system presented in chapter 5. In section 7.4.2 we will examine the performance of the protocol and prove a result giving the probability of a message being correctly transmitted within a certain time. It will turn out that this latter proof will make use of many results proved during the former proof: in section 7.4.1 we will prove many results about the ordering of events that will prove useful in section 7.4.2.

We introduce a piece of notation which will be useful in the proofs. We want to be able to talk about the order in which events occur without mentioning the times explicitly. We shall write untimed u, where u is a sequence of untimed events, to specify that events are performed in the order given by u:

$$(\mathsf{untimed}\quad u)(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} \forall s' \in \mathit{tstrip}\ s \quad s' \quad u$$

where $\mathit{tstrip}\ s$ returns the set of all sequences of untimed events corresponding to the trace s, and is the prefix relation on traces.

7.4.1 Safety

We begin by showing that the protocol acts like a one place buffer:

Theorem 7.4.1: $PROT\ \mathbf{sat}_\rho\ S$ where

$$S \mathrel{\widehat{=}} \exists\, n : \quad ; x_1, \ldots, x_n : X \quad \mathsf{untimed} \quad \langle in.x_1, out.x_1, \ldots, in.x_n, out.x_n \rangle$$

♡

Proof: We use the proof rule for hiding (rule B.1.25) to reduce the proof obligation to showing that $PROT'$ satisfies the predicate

$$\mathsf{internal}\ A \Rightarrow \exists\, n : \quad ; x_1, \ldots, x_n : X \quad \mathsf{untimed}\ \{in, out\} \quad \langle in.x_1, out.x_1, \ldots, in.x_n, out.x_n \rangle$$

Note that the predicate on the right hand side of the implication is A-independent and implies S. This predicate can be strengthened to

$$S' \mathrel{\widehat{=}} \mathsf{internal}\ A \Rightarrow \begin{pmatrix} rm.x\ \mathsf{at}\ t \Rightarrow lm.x\ \mathsf{at}\ t - t_0 \\ \wedge\ \exists\, n : \quad ; x_1, \ldots, x_n : X\,; n_1, \ldots, n_n : \quad ^+ \quad \mathsf{untimed} \quad s_1 \quad s_2 \quad .. \quad s_n \\ \quad \text{where } s_i = \langle in.x_i \rangle \quad \langle lm.x_i \rangle^{n_i} \quad \langle rm.x_i, conf_1, out.x_i, conf_2 \rangle \end{pmatrix}$$

All rms must have been caused by an lm t_0 time units previously; the protocol repeatedly performs the events in, one or more lms, rm, $conf_1$, out, $conf_2$, in that order.

We now seek specifications for M and $S \underset{conf}{\Vert} R$. Let S_M and S_{SR} be given by

$$\begin{aligned} S_M \ \mathrel{\widehat{=}}\ & rm.x\ \mathsf{at}\ t \Rightarrow \mathsf{name\ of\ last\ before}\ t = lm.x \\ & \wedge\ lm.x\ \mathsf{at}\ t \Rightarrow (rm.x\ \mathsf{live\ from}\ t + t_0 \vee \mathsf{no}\ rm\ \mathsf{offered}\ (t, \mathsf{time\ of\ first\ after}\ t]) \\ & \wedge\ lm, rm\ \mathsf{separate} \end{aligned}$$

$$S_{SR} \;\widehat{=}\; \begin{pmatrix} \text{internal } conf \\ \wedge\ rm.x \text{ at } t \Rightarrow \text{name of last } lm, rm \text{ before } t = lm.x \\ \wedge\ lm.x \text{ at } t \Rightarrow \begin{pmatrix} rm.x \text{ accessible from } t + t_0 \\ \vee \text{ no } rm \text{ at } (t, \text{time of first } lm, rm \text{ after } t] \end{pmatrix} \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} rm.x \text{ at } t \Rightarrow lm.x \text{ at } t - t_0 \\ \wedge\ \exists\, n : \quad ; x_1, \ldots, x_n : X \,;\, n_1, \ldots, n_n : \quad ^{+} \quad \text{untimed} \quad s_1 \quad s_2 \quad \ldots \quad s_n \\ \quad \text{where } s_i = \langle in.x_i \rangle \quad \langle lm.x_i \rangle^{n_i} \quad \langle rm.x_i, conf_1, out.x_i, conf_2 \rangle \end{pmatrix}$$

$$\wedge\ rm, conf, out \text{ separate}$$

$$\wedge\ lm \text{ at } t \wedge in \text{ live from } t' > t \Rightarrow$$

$$\exists\, t_1, t_2, t_3 \quad t < t_1 < t_2 < t_3 < t' \wedge conf_1 \text{ at } t_1 \wedge out \text{ at } t_2 \wedge conf_2 \text{ at } t_3$$

The three conjuncts of S_M state that

- *rm*s must be preceded by corresponding *lm*s;
- if an *lm* is performed, then either an *rm* will be offered until it is performed, or no *rm* will be performed until after another *lm* — i.e. the message is either correctly transmitted or lost; *and*
- the medium offers *lm*s and *rm*s separately.

The first conjunct of S_{SR} states that if the environment is such that

- *conf* is always available;
- all *rm*s are preceded by corresponding *lm*s; *and*
- after an *lm* occurs, an *rm* is either offered until accepted or not offered at all;

then

- each *rm* occurs t_0 after a corresponding *lm*; *and*
- the trace is of the required form.

The second and third conjuncts say that at most one of *rm*, *conf* and *out* are offered at a time, and that $conf_1$, *out* and $conf_2$ occur between each *lm* and *in*.

We want to reduce our proof obligation to proving that $M \;\mathbf{sat}_\rho\; S_M$ and $S \underset{conf}{\Vert} R \;\mathbf{sat}_\rho\; S_{SR}$. From the proof rule for parallel composition, it is enough to prove the following:

Lemma 7.4.1.1: If $\Sigma s \subseteq \{in, lm, rm, conf, out\}$ then

$$S_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\}) \wedge S_{SR}(\tau, \sqsubseteq_{SR}, s) \Rightarrow S'(\tau, \sqsubseteq_{SR} \underset{lm,rm}{\Vert} \sqsubseteq_M, s)$$

♡

Proof of lemma: Let $\sqsubseteq \;=\; \sqsubseteq_{SR} \underset{lm,rm}{\Vert} \sqsubseteq_M$. Suppose the premises of the lemma hold, and suppose internal $A(\tau, \sqsubseteq, s)$. We will aim to prove

$$\begin{pmatrix} rm.x \text{ at } t \Rightarrow lm.x \text{ at } t - t_0 \\ \wedge\ \exists\, n : \quad ; x_1, \ldots, x_n : X \,;\, n_1, \ldots, n_n : \quad ^{+} \quad \text{untimed} \quad s_1 \quad s_2 \quad \ldots \quad s_n \\ \quad \text{where } s_i = \langle in.x_i \rangle \quad \langle lm.x_i \rangle^{n_i} \quad \langle rm.x_i, conf_1, out.x_i, conf_2 \rangle \end{pmatrix} (\tau, \sqsubseteq, s)$$

by induction on tbe number of events in s.

Suppose then that the lemma holds for all traces of length less than l, and let $\#s = l$. We begin by showing

$$\left(\begin{array}{l}\text{internal } conf \\ \wedge\ rm.x \text{ at } t \Rightarrow \text{name of last } lm, rm \text{ before } t = lm.x \\ \wedge\ lm.x \text{ at } t \Rightarrow \left(\begin{array}{l} rm.x \text{ accessible from } t + t_0 \\ \vee \text{ no } rm \text{ at } (t, \text{time of first after } t] \end{array}\right)\end{array}\right)(\tau, \sqsubseteq_{SR}, s) \qquad (*)$$

The crux is the third conjunct, for which we need the inductive hypothesis. $(*)$ matches the left hand side of the first conjunct of S_{SR}, which will enable us to deduce the result.

The first conjunct of $(*)$ follows immediately from the assumption and the second conjunct follows from the first conjunct of S_M. For the third conjunct, suppose $(lm.x \text{ at } t)(\tau, \sqsubseteq_{SR}, s)$. Then the second conjunct of S_M gives us

$$(rm.x \text{ live from } t + t_0 \vee \text{no } rm \text{ offered } (t, \text{time of first after } t])(\tau, \sqsubseteq_M, s \quad \{in, out\})$$

If $(\text{no } rm \text{ offered } (t, \text{time of first after } t])(\tau, \sqsubseteq_M, s \quad \{in, out\})$ then we have $(\text{no } rm \text{ at } (t, \text{time of first after } t])(\tau, \sqsubseteq_{SR}, s)$, as required. So suppose $(rm.x \text{ live from } t + t_0)(\tau, \sqsubseteq_M, s \quad \{in, out\})$; we will show $(rm.x \text{ accessible from } t + t_0)(\tau, \sqsubseteq_{SR}, s)$. Taking $c = rm$, $C = \{rm\}$ in corollary 5.3.2, we need to show

$$(\text{internal } rm)(\tau, \sqsubseteq, s) \tag{7.1}$$

$$rm, \{in, out, conf\} \text{ separate from } t + t_0 \tag{7.2}$$

(7.1) follows from the hypothesis internal A. To show (7.2), suppose for some $t' \quad t + t_0$ we have $(t', rm) \in \text{items} \sqsubseteq$; then from the second conjunct of S_{SR} we have $(t', out), (t', conf) \notin \text{items} \sqsubseteq$. It remains to show $(t', in) \notin \text{items} \sqsubseteq$. Suppose otherwise: in this case, the rm must bave been caused by au earlier lm, but for S to be willing to perform in, there must have beeu a $conf_1, out, conf_2$ in between the lm and in (by the third conjunct of S_{SR}):

$$\begin{array}{rllllll} S \underset{conf}{\Vert\!\!\!\!\!\leftarrow} R: & \ldots & lm & conf_1 & out & conf_2 & in \\ M: & \ldots & lm & & & & rm \end{array}$$

Suppose the *out* occurs at time τ'. Consider the behaviours $(\tau', \sqsubseteq_M \quad \tau', s \quad \{lm, rm\} \quad \tau')$ and $(\tau', \sqsubseteq_{SR} \quad \tau', s \quad \tau')$. These satisfy S_M and S_{SR} respectively, so by the inductive hypothesis the behaviour of the composite process, $(\tau', \sqsubseteq \quad \tau', s \quad \tau')$, should satisfy the result of the lemma. However, it clearly doesn't as a $conf_1$ follows an lm. Hence we reach a contradiction and so conclude that (7.2) holds, and so $(rm.x \text{ accessible from } t + t_0)(\tau, \sqsubseteq_{SR}, s)$.

Hence we have proved $(*)$ and so can use S_{SR} to deduce

$$\left(\begin{array}{l} rm.x \text{ at } t \Rightarrow lm.x \text{ at } t - t_0 \\ \wedge\ \exists\, n : \quad ; x_1, \ldots, x_n : X\, ; n_1, \ldots, n_n : \quad ^{+} \quad \text{untimed} \quad s_1 \quad s_2 \quad \ldots \quad s_n \\ \qquad \text{where } s_i = \langle in.x_i \rangle \quad \langle lm.x_i \rangle^{n_i} \quad \langle rm.x_i, conf_1, out.x_i, conf_2 \rangle \end{array}\right)(\tau, \sqsubseteq_{SR}, s)$$

and heuce

$$\left(\begin{array}{l} rm.x \text{ at } t \Rightarrow lm.x \text{ at } t - t_0 \\ \wedge\ \exists\, n : \quad ; x_1, \ldots, x_n : X\, ; n_1, \ldots, n_n . \quad ^{+} \quad \text{untimed} \quad s_1 \quad s_2 \quad \ldots \quad s_n \\ \qquad \text{where } s_i = \langle in.x_i \rangle \quad \langle lm.x_i \rangle^{n_i} \quad \langle rm.x_i, conf_1, out.x_i, conf_2 \rangle \end{array}\right)(\tau, \sqsubseteq, s)$$

as required. □

We now prove that M and $S \underset{conf}{\Vert} R$ satisfy their specifications. We start by proving $M \ \mathbf{sat}_\rho \ S_M$. We assume $X \ \mathbf{sat}_\rho \ S_M$ and use the proof rule for recursion to reduce the proof obligation to

$$lm?x \xrightarrow{t_0} (rm!x \longrightarrow X \ {}_p\sqcap_q \ WAIT \ \delta \, ; X) \ \mathbf{sat}_\rho \ S_M$$

Using the proof rule for prefixing, it is enough to prove that

$$\begin{aligned} rm!x \longrightarrow X \ {}_p\sqcap_q \ WAIT \ \delta \, ; X \ \mathbf{sat}_\rho \ & (rm.x \text{ live from } 0 \vee \text{no } rm \text{ at } [0, \text{time of first}]) \\ & \wedge \text{ name of first} = rm.y \Rightarrow y = x \\ & \wedge \begin{pmatrix} rm.x \text{ at } t \\ \wedge \ t > \text{time of first} \end{pmatrix} \Rightarrow \text{name of last before } t = lm.x \\ & \wedge \ lm.x \text{ at } t \Rightarrow \begin{pmatrix} rm.x \text{ live from } t + t_0 \\ \vee \text{ no } rm \text{ offered } (t, \text{time of first after } t] \end{pmatrix} \\ & \wedge \ lm, rm \text{ separate} \end{aligned}$$

We now use the proof rule for probabilistic choice to reduce the proof obligation to

$$\begin{aligned} rm!x \longrightarrow X \ \mathbf{sat}_\rho \ & rm.x \text{ live from } 0 \\ & \wedge \text{ name of first} = rm.y \Rightarrow y = x \\ & \wedge \ rm.x \text{ at } t > \text{time of first} \Rightarrow \text{name of last before } t = lm.x \\ & \wedge \ lm.x \text{ at } t \Rightarrow (rm.x \text{ live from } t + t_0 \vee \text{no } rm \text{ offered } (t, \text{time of first after } t]) \\ & \wedge \ lm, rm \text{ separate} \end{aligned}$$

and

$$\begin{aligned} WAIT \ \delta ; X \ \mathbf{sat}_\rho \ & \text{no } rm \text{ at } [0, \text{time of first}] \\ & \wedge \text{ name of first} = rm.y \Rightarrow y = x \\ & \wedge \ rm.x \text{ at } t > \text{time of first} \Rightarrow \text{name of last before } t = lm.x \\ & \wedge \ lm.x \text{ at } t \Rightarrow (rm.x \text{ live from } t + t_0 \vee \text{no } rm \text{ offered } (t, \text{time of first after } t]) \\ & \wedge \ lm, rm \text{ separate} \end{aligned}$$

These are easily proven using the proof rules for prefixing and delay, making use of the assumption about X.

We now turn our attention to proving that $S \underset{conf}{\Vert} R \ \mathbf{sat}_\rho \ S_{SR}$. We seek specifications for S and R. Let

$$\begin{aligned} S_S \ \widehat{=} \ & \exists \, n : \quad ; x_1, \ldots, x_n : X \, ; n_1, \ldots, n_n : \quad ^{+} \quad \text{untimed} \quad s_1 \quad s_2 \quad \ldots \quad s_n \\ & \qquad \text{where } s_i = \langle in.x_i \rangle \quad \langle lm.x_i \rangle^{n_i} \quad \langle conf_1, conf_2 \rangle \\ & \wedge \ lm.x \text{ at } t \Rightarrow conf_1 \text{ live } [t + \delta, t + t_0 + \delta] \wedge \text{no } lm \text{ offered } (t, t + t_0 + \delta] \end{aligned}$$

$$\begin{aligned} S_R \ \widehat{=} \ & rm?x \text{ at } t \Rightarrow conf_1 \text{ live from } t + \delta \\ & \wedge \ conf_2 \text{ at } t \Rightarrow rm \text{ live from } t + \delta \\ & \wedge \ rm \text{ live from } 0 \\ & \wedge \ \exists \, n : \quad ; x_1, \ldots, x_n : X \quad \text{untimed} \quad s_1 \quad s_2 \quad \ldots \quad s_n \\ & \qquad \text{where } s_i = \langle rm.x_i, conf_1, out.x_i, conf_2 \rangle \\ & \wedge \ rm, conf, out \text{ separate} \end{aligned}$$

We have the following proof obligation:

Lemma 7.4.1.2: If $\Sigma s \subseteq \{lm, rm, in, out, conf\}$ then

$$S_S(\tau, \sqsubseteq_S, s \quad A_S) \wedge S_R(\tau, \sqsubseteq_R, s \quad A_R) \Rightarrow S_{SR}(\tau, \sqsubseteq_S \underset{conf}{\Vert} \sqsubseteq_R, s)$$

$\heartsuit$

Proof of lemma: The second conjunct of S_{SR} follows from the last conjunct of S_R; the third conjunct of S_{SR} follows from the first conjunct of S_S and the fourth conjunct of S_R. We concentrate on proving the first conjunct of S_{SR}.
Let $\sqsubseteq \mathrel{\hat{=}} \sqsubseteq_S \underset{conf}{\Vert} \sqsubseteq_R$. Assume the premises of the lemma, and suppose

$$\begin{pmatrix} \text{internal } conf \\ \wedge\ rm.x \text{ at } t \Rightarrow \text{name of last } lm, rm \text{ before } t = lm.x \\ \wedge\ lm.x \text{ at } t \Rightarrow \begin{pmatrix} rm.x \text{ accessible from } t + t_0 \\ \vee \text{ no } rm \text{ at } (t, \text{time of first } lm, rm \text{ after } t] \end{pmatrix} \end{pmatrix} (\tau, \sqsubseteq, s) \qquad (*)$$

We want to prove

$$\begin{pmatrix} rm.x \text{ at } t \Rightarrow lm.x \text{ at } t - t_0 \\ \wedge\ \exists\, n : \quad ; x_1, \ldots, x_n : X\ ; n_1, \ldots, n_n : \quad ^{+} \quad \text{untimed} \quad s_1 \quad s_2 \quad \ldots \quad s_n \\ \quad \text{where } s_i = \langle in.x_i \rangle \quad (lm.x_i)^{n_i} \quad \langle rm.x_i, conf_1, out.x_i, conf_2 \rangle \end{pmatrix} (\tau, \sqsubseteq, s)$$

We concentrate on the second conjunct; the first conjunct is proved in passing. We consider in what order the parallel composition performs events.
The first event (if it performs any events) of $s \quad A_S$ is $in.x_1$ for some x_1 and the first event of $s \quad A_R$ is $rm.x$ for some x. But from $(*)$ we have $rm.x$ at $t \Rightarrow$ name of last lm, rm before $t = lm.x$ so $rm.x$ cannot be the first event. Hence the first event must be $in.x_1$ for some x_1.
We consider now the identity of the second event (if there is a second event). The next event of $s \quad A_S$ is $lm.x_1$, while the next event of $s \quad A_R$ is $rm.x$ for some x. But, as in the previous paragraph, an rm cannot yet be performed. Hence the second event is $lm.x_1$.
Suppose the process performs $lm.x_i$ at some time t, and an rm has not yet been performed. Then from $(*)$ we have

$$(rm.x \text{ accessible from } t + t_0 \vee \text{no } rm \text{ at } (t, \text{time of first } lm, rm \text{ after } t])(\tau, \sqsubseteq, s)$$

We consider these two disjuncts separately.

- If the first disjunct holds, then since $(rm \text{ live from } 0)(\tau, \sqsubseteq_R, s \quad A_R)$ we have $rm.x$ at $t + t_0$. From the second conjunct of S_S we have no lm at $(t, t_0]$ so the next event is $rm.x$. Note that this rm occurs t_0 after a corresponding lm, as required by the first clause of our desired result.
- If no rm at $(t, \text{time of first } lm, rm \text{ after } t])(\tau, \sqsubseteq, s)$ then the next event must be another $lm.x$, by the first clause of S_S and the fourth clause of S_R.

Suppose then that the first $rm.x$ occurs at some time t. Then from the previous paragraph, we must have had $lm.x$ at $t - t_0$. So from the first conjunct of S_R we have ($conf_1$ live from $t + \delta)(\tau, \sqsubseteq_R, s \;\; A_R)$. Also from the second conjunct of S_S we have ($conf_1$ live $[t - t_0 + \delta, t + \delta] \wedge$ no lm at $(t - t_0, t + \delta])(\tau, \sqsubseteq_S, s \;\; A_S)$ so the next event is $conf_1$ which occurs at time $t + \delta$, since we have internal $conf$ by assumption.

Now suppose that a $conf_1$ occurs at some time and the previous event was $rm.x$. Then the next two events of $s \;\; A_R$ are $out.x$ and $conf_2$ and the next event of $s \;\; A_S$ is $conf_2$, so the next two events are $out.x$ and $conf_2$.

Now suppose a $conf_2$ occurs at some time. Then the next event of $s \;\; A_R$ is $rm.x$ for some x, and the next two events of $s \;\; A_S$ are $in.x$ and $lm.x$ for some x. But as above, the $rm.x$ cannot occur before the lm, so the next two events are $in.x$ and $lm.x$.

Suppose now that an $lm.x$ occurs at time t, and an rm has occurred previously. Then from the above a $conf_2$ must have occurred at some time t', before t. Then from the second conjunct of S_R we have (rm live from $t' + \delta)(\tau, \sqsubseteq_R, s \;\; A_R)$, so ($rm$ live from $t + \delta)(\tau, \sqsubseteq_R, s \;\; A_R)$. Also, from (*) we have

$$(rm.x \text{ accessible from } t + t_0 \vee \text{no } rm \text{ at } (t, \text{time of first } lm, rm \text{ after } t])(\tau, \sqsubseteq, s)$$

Then as above, we either have the next event an $rm.x$ at time $t + t_0$, or the next event is another $lm.x$. Note that in the case where the $rm.x$ occurs, it happens t_0 after a corresponding lm, as required by the first clause of our desired result.

Finally, suppose an $rm.x$ occurs at time t and this is not the first rm. Then from the previous paragraph, we must have had an lm at $t - t_0$. So, as above, the next event must be a $conf_1$.

□

It remains to show that S and R meet their specifications. These are easily shown using the proof rules for recursion, prefixing and time out. This completes the proof. □

7.4.2 Liveness

We will now consider the liveness properties of the protocol. We want to calculate the probability of a message being correctly transmitted within a certain time. For $n \in$ let

$$T \mathrel{\hat=} t_0 + 2\delta \qquad T_n \mathrel{\hat=} nT + \delta$$

T is the time between successive attempts at transmission; T_n is the time taken for the message to get through if the nth attempt at transmission is successful. Let G be the predicate that an x is input at time t; let S_n be the predicate that the output occurs within time T_n (for $n > 0$):

$$G \mathrel{\hat=} in.x \text{ at } t \qquad S_n \mathrel{\hat=} out.x \text{ live from}(t, t + T_n]$$

where we overload the live from construct to specify that an event becomes available at some time during an interval:

$$a \text{ live from } I \mathrel{\hat=} \exists\, t \in I \quad a \text{ live from } t$$

Informally, S_n is true if the message is transmitted within n attempts. We want to prove (for all t and x):

Theorem 7.4.2: $\forall\, n : \;^{+} \;\; PROT\ \mathbf{sat}_p^{\geq 1 - q^n}\ S_n \mid G$. ♡

Proof: We fix n and attempt to prove $PROT\ \mathbf{sat}_\rho^{\geqslant 1-q^n}\ S_n \mid G$. We consider the case where the message is correctly transmitted on the mth attempt; for $m \in\ ^+$, let

$$S'_m \mathrel{\widehat{=}} \textsf{no}\ out\ \textsf{offered}\ (t, t + T_{m-1}] \land out.x\ \textsf{live from}(t + T_{m-1}, t + T_m]$$

S'_m is the condition that the message is correctly transmitted on the mth attempt. We use rule 7.1.19 to reduce our proof obligation to

$$\forall m \quad PROT\ \mathbf{sat}_\rho^{\geqslant p \cdot q^{m-1}}\ S'_m \mid G$$

using $S'_1 \ldots S'_n$ to prove S_n. We have the following proof obligations:

- $\forall m \quad n \quad S'_m(\tau, \sqsubseteq, s) \land G(\tau, \sqsubseteq, s) \Rightarrow S_n(\tau, \sqsubseteq, s) \land G(\tau, \sqsubseteq, s)$;
- $(S'_1(\tau, \sqsubseteq, s) \land G(\tau, \sqsubseteq, s), \ldots, S'_n(\tau, \sqsubseteq, s) \land G(\tau, \sqsubseteq, s))$ *disjoint*;
- $\sum_{m=1}^{n} p.q^{m-1} \quad 1 - q^n$.

These are all trivial.

We fix $m \in\ ^+$ and seek to prove $PROT\ \mathbf{sat}_\rho^{\geqslant p \cdot q^{m-1}}\ S'_m \mid G$. We use the proof rule for hiding to reduce our proof obligation to $PROT'\ \mathbf{sat}_\rho^{\geqslant p \cdot q^{m-1}}\ S''_m \mid G$ where

$$S''_m \mathrel{\widehat{=}} \textsf{internal}\ A \Rightarrow \textsf{no}\ out\ \textsf{offered}\ (t, t + T_{m-1}] \land out.x\ \textsf{live from}(t + T_{m-1}, t + T_m]$$

To prove this specification, we introduce the following specification (for $l \in$):

$$S'''_l \mathrel{\widehat{=}} \textsf{internal}\ A \Rightarrow \textsf{no}\ out\ \textsf{offered}\ (t, t + T_l] \land (\textsf{beyond}\ t + T_l \Rightarrow lm.x\ \textsf{at}\ t + T_l)$$

S'''_l is the condition that the first l attempts at transmission are unsuccessful, and the protocol tries transmitting for the $l + 1$th time at $t + T_l$. We use rule 7.1.22 to reduce the proof obligation to the following:

1. $PROT'\ \mathbf{sat}_\rho^{\geqslant 1}\ S'''_0 \mid G$;
2. $\forall l : \quad PROT'\ \mathbf{sat}_\rho^{\geqslant q}\ S'''_{l+1} \mid S'''_l \land G$;
3. $PROT'\ \mathbf{sat}_\rho^{\geqslant p}\ S''_{m+1} \mid S'''_m \land G$.

We prove each of these in turn.

Condition 1: We can rule 7.1.13 to reduce the proof obligation to

$$PROT'\ \mathbf{sat}_\rho\ in.x\ \textsf{at}\ t \Rightarrow \textsf{no}\ out\ \textsf{offered}\ (t, t + \delta] \land lm.x\ \textsf{live from}\ t + \delta$$

By the results of section 7.4.1, if an *in* occurs at time t, then *out* does not occur during $(t, t + \delta]$, and the medium must be ready to receive an *lm* from time $t + \delta$, so we can use the proof rule for parallel composition to reduce the proof obligation to

$$S \underset{conf}{\Vert}\ R\ \mathbf{sat}_\rho\ in.x\ \textsf{at}\ t \Rightarrow lm.x\ \textsf{live from}\ t + \delta$$

We can use the proof rule for parallel composition again to reduce the proof obligation to

$$S\ \mathbf{sat}_\rho\ in.x\ \textsf{at}\ t \Rightarrow lm.x\ \textsf{live from}\ t + \delta \qquad \text{and} \qquad R\ \mathbf{sat}_\rho\ true$$

The proof obligation for R is trivial; the proof obligation for S can be discharged using the proof rules for prefixing and recursion. □

Condition 2: We must prove $\forall l : \quad PROT' \ \mathbf{sat}_\rho^{\geqslant q} \ S'''_{l+1} \mid S'''_l \wedge G$. Pick $l \in \quad$. The condition $S'''_{l+1} \mid S'''_l \wedge G$ is equivalent to

$$\left.\begin{array}{l}\mathsf{internal}\, A \Rightarrow \\ \quad\begin{pmatrix}\mathsf{no}\ out\ \mathsf{offered}\ (t, t + T_{l+1}] \\ \wedge\ \mathsf{beyond}\, t + T_{l+1} \Rightarrow lm.x\ \mathsf{at}\ t + T_{l+1}\end{pmatrix}\end{array}\right| \begin{array}{l}\mathsf{internal}\, A \Rightarrow \\ \quad\begin{pmatrix}\mathsf{no}\ out\ \mathsf{offered}\ (t, t + T_l] \\ \wedge\ \mathsf{beyond}\, t + T_l \Rightarrow lm.x\ \mathsf{at}\ t + T_l\end{pmatrix} \\ \wedge\ in.x\ \mathsf{at}\ t\end{array}$$

which is the condition that the first $l + 1$ attempts are unsuccessful given that the first l are unsuccessful. By rule 7.1.16 we can reduce this to $S'''_{l+1} \mid G_l$ where

$$G_l \mathrel{\widehat{=}} \mathsf{no}\ out\ \mathsf{offered}\ (t, t + T_l] \wedge (\mathsf{beyond}\, t + T_l \Rightarrow lm.x\ \mathsf{at}\ t + T_l) \wedge in.x\ \mathsf{at}\ t$$

We will reduce the proof obligation to

$$S \underset{conf}{\|\!|\!\|} R\ \mathbf{sat}_\rho^{\geqslant 1}\ S_{SR} \mid G_{SR} \quad \text{and} \quad M\ \mathbf{sat}_\rho^{\geqslant q}\ S_M \mid G_M$$

where

$$\begin{aligned}
S_{SR} &\mathrel{\widehat{=}} \mathsf{internal}\ conf \wedge \mathsf{no}\ rm\ \mathsf{at}\ [t + T_l, t + T_l + t_0] \Rightarrow \\
&\qquad \mathsf{no}\ out\ \mathsf{offered}\ (t, t + T_{l+1}] \wedge lm.x\ \mathsf{live\ from}\ t + T_{l+1} \\
G_{SR} &\mathrel{\widehat{=}} \mathsf{no}\ out\ \mathsf{offered}\ (t, t + T_l] \wedge (\mathsf{beyond}\, t + T_l \Rightarrow lm.x\ \mathsf{at}\ t + T_l) \wedge in.x\ \mathsf{at}\ t \\
S_M &\mathrel{\widehat{=}} lm.x\ \mathsf{at}\ t + T_l \Rightarrow \mathsf{no}\ rm\ \mathsf{offered}\ [t + T_l, t + T_l + t_0] \wedge lm\ \mathsf{live\ from}\ t + T_l + t_0 + \delta \\
G_M &\mathrel{\widehat{=}} true
\end{aligned}$$

$S_{SR} \mid G_{SR}$ is the condition that, given that an lm occurs at $t + T_l$, if an rm does not occur within the next t_0, then an lm is offered at time $t + T_{l+1}$, i.e. the condition that the protocol tries to retransmit as it should. $S_M \mid G_M$ is the condition that the medium loses a message that is input at time $t + T_l$, and then becomes ready for another input.

We have the following proof obligation:

Lemma 7.4.2.1: If $\Sigma s \subseteq \{in, out, lm, rm, conf\}$ and $\sqsubseteq\ =\ \sqsubseteq_{SR} \underset{lm,rm}{\|\!|\!\|} \sqsubseteq_M$ then

$$\begin{pmatrix}S_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge G_{SR}(\tau, \sqsubseteq_{SR}, s) \\ \wedge\ S_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\}) \wedge G_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\})\end{pmatrix} \Rightarrow S'''_{l+1}(\tau, \sqsubseteq, s) \wedge G_l(\tau, \sqsubseteq, s)$$

and

$$G_l(\tau, \sqsubseteq, s) \Rightarrow G_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge G_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\})$$

♡

Proof of lemma: For the first obligation assume

$$S_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge G_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge S_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\}) \wedge G_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\})$$

Then G_l follows immediately from G_{SR}. To prove S'''_{l+1}, assume $(\mathsf{internal}\, A)(\tau, \sqsubseteq, s)$. We must show $(\mathsf{no}\ out\ \mathsf{offered}\ (t, t + T_{l+1}] \wedge (\mathsf{beyond}\, t + T_{l+1} \Rightarrow lm.x\ \mathsf{at}\ t + T_{l+1}))(\tau, \sqsubseteq, s)$.

If $\neg$ beyond $t + T_l$ then the result is trivial from the first conjunct of G_{SR}. So suppose beyond $t + T_l$; then from the second conjunct of G_{SR} we have $lm.x$ at $t + T_l$. Then from S_M we have no rm at $[t + T_l, t + T_l + t_0]$, so S_{SR} gives us (no out offered $(t, t + T_{l+1}] \wedge lm.x$ live from $t + T_{l+1})(\tau, \sqsubseteq_{SR}, s)$. Also, S_M gives us (lm live from $t + T_l + t_0 + \delta)(\tau, \sqsubseteq_M, s \;\; \{lm, rm\})$, and so we have

$$(\text{no } out \text{ offered } (t, t + T_{l+1}] \wedge (\text{beyond } t + T_{l+1} \Rightarrow lm.x \text{ at } t + T_{l+1}))(\tau, \sqsubseteq, s)$$

from the assumption (internal $A)(\tau, \sqsubseteq, s)$ and the definition of parallel composition of offer relations.

For the second obligation, assume $G_l(\tau, \sqsubseteq, s)$. Then $G_M(\tau, \sqsubseteq_M, s \;\; \{lm, rm\})$ is trivially true and $G_{SR}(\tau, \sqsubseteq_{SR}, s)$ follows immediately from the assumption. □

We now prove that $S \underset{conf}{\Vert} R$ satisfies its specification: $S \underset{conf}{\Vert} R$ $\mathbf{sat}_\rho^{\geqslant l}$ $S_{SR} \mid G_{SR}$. By rule 7.1.13, we can reduce the proof obligation to $S \underset{conf}{\Vert} R$ $\mathbf{sat}_\rho$ $(G_{SR} \Rightarrow S_{SR})$. The predicate $G_{SR} \Rightarrow S_{SR}$ can be strengthened to $S'_{SR}(t + T_l)$ where

$$\begin{aligned} S'_{SR}(t') \mathrel{\widehat{=}} \text{internal } conf \wedge lm.x \text{ at } t' \wedge \text{no } rm \text{ at } [t', t' + t_0] \Rightarrow \\ \text{no } out \text{ offered } (t', t' + T] \wedge lm.x \text{ live from } t' + T \end{aligned}$$

We show that $S'_{SR}(t')$ is met for *all* t'. We reduce the proof obligations to proving that S $\mathbf{sat}_\rho$ S_S and R $\mathbf{sat}_\rho$ S_R where the predicates S_S and S_R are given by

$$\begin{aligned} S_S &\mathrel{\widehat{=}} lm.x \text{ at } t' \wedge \text{no } conf_l \text{ at } [t', t' + t_0 + \delta] \Rightarrow lm.x \text{ live from } t' + T \\ S_R &\mathrel{\widehat{=}} \text{no } rm \text{ at [time of last } conf \text{ before } t'', t'' - \delta] \Rightarrow \text{no } conf_l \text{ at } t'' \end{aligned}$$

S's specification says that if it does not receive a $conf_l$ within $t_0 + \delta$ of performing an lm, then it tries retransmitting. R's specification says that if a $conf$ occurs then it cannot perform another $conf_l$ until at least δ after an rm occurs.

We have the following proof obligation:

Lemma 7.4.2.2:

$$S_S(\tau, \sqsubseteq_S, s \;\; \{in, lm, conf\}) \wedge S_R(\tau, \sqsubseteq_R, s \;\; \{rm, conf, out\}) \Rightarrow S'_{SR}(\tau, \sqsubseteq_S \underset{conf}{\Vert} \sqsubseteq_R, s)$$

♡

Proof of lemma: Let $\sqsubseteq \mathrel{\widehat{=}} \sqsubseteq_S \underset{conf}{\Vert} \sqsubseteq_R$ and assume the premises. Suppose

$$(\text{internal } conf \wedge lm.x \text{ at } t' \wedge \text{no } rm \text{ at } [t', t' + t_0])(\tau, \sqsubseteq, s)$$

We will show (no out offered $(t', t' + T] \wedge lm.x$ live from $t' + T)(\tau, \sqsubseteq, s)$. From the results of section 7.4.1 we have no rm at [time of last $conf$ before $t', t']$, and from the assumption we have no rm at $[t', t' + t_0]$; hence for all $t'' \in [t', t' + t_0 + \delta]$ we have no rm at [time of last $conf$ before $t'', t'' - \delta]$, so from S_R we have no $conf_l$ at t''. Hence we have no $conf_l$ at $[t', t' + t_0 + \delta]$, so from S_S we have $lm.x$ live from $t' + T$. Also, from the results of section 7.4.1 we have out offered $t'' \Rightarrow rm$ at (time of last $lm, t'' - 2\delta]$, hence no out offered $(t', t' + T]$, as reqnired. □

The proofs that S and R satisfy their specifications are completely routine.

We now prove that M satisfies its specification: $M\ \mathbf{sat}_\rho^{\geqslant q}\ S_M \mid G_M$. Since $G_M = \mathit{true}$, the proof obligation can be reduced to $M\ \mathbf{sat}_\rho^{\geqslant q}\ S'_M(t + T_l)$ where

$$S'_M(t') \mathrel{\widehat{=}} lm.x \text{ at } t' \Rightarrow \text{no } rm \text{ offered } [t', t' + t_0] \wedge lm \text{ live from } t' + t_0 + \delta$$

We use the proof rule for recursion to show $\forall t' \quad M\ \mathbf{sat}_\rho^{\geqslant q}\ S'_M(t')$. Note that $STOP\ \mathbf{sat}_\rho$ $S'_M(t')$ so $S'_M(t')$ is satisfiable (so the side condition of the proof rule for recursion is satisfied). We assume $\forall t' \quad X\ \mathbf{sat}_\rho^{\geqslant q}\ S'_M(t')$; we must show $\forall t' \quad lm?x \xrightarrow{t_0} (rm!x \longrightarrow X\ {}_p\sqcap_q\ WAIT\ \delta\,;$ $X)\ \mathbf{sat}_\rho^{\geqslant q}\ S'_M(t')$. The following result about M is trivial to prove and will be useful:

$$M\ \mathbf{sat}_\rho\ lm \text{ live from } 0 \qquad (*)$$

Pick t' and suppose $lm.x$ at t'. We have two cases to consider.

- If the lm at time t' is the *first* lm then we use the proof rule for prefixing to reduce the proof obligation to

$$rm!x \longrightarrow X\ {}_p\sqcap_q\ WAIT\ \delta\,; X\ \mathbf{sat}_\rho^{\geqslant q}\ S''_M$$
$$\text{where } S''_M \mathrel{\widehat{=}} \text{no } rm \text{ offered } 0 \wedge lm \text{ live from } \delta$$

  We can then use the proof rule for probabilistic choice to reduce the obligation to

$$rm!x \longrightarrow X\ \mathbf{sat}_\rho^{\geqslant 0}\ S''_M \quad \text{and} \quad WAIT\ \delta\,; X\ \mathbf{sat}_\rho^{\geqslant q}\ S''_M$$

  The first obligation follows from rule 7.1.5; the second obligation follows from $(*)$ and the rule for delay.

- If the first lm occurs at some time $t'' < t'$ then we can use the proof rule for prefixing to reduce the proof obligation to

$$rm!x \longrightarrow X\ {}_p\sqcap_q\ WAIT\ \delta\,; X\ \mathbf{sat}_\rho^{\geqslant q}\ S'_M(t' - t'' - t_0)$$

  The proof rule for probabilistic choice can then be used to reduce this to

$$rm!x \longrightarrow X\ \ \mathbf{sat}_\rho^{\geqslant q}\ \ S'_M(t' - t'' - t_0) \qquad (7.3)$$
$$WAIT\ \delta\,; X\ \ \mathbf{sat}_\rho^{\geqslant q}\ \ S'_M(t' - t'' - t_0) \qquad (7.4)$$

  For (7.3), suppose the first rm occurs after a delay of t'''; then the proof rule for prefixing can be used to reduce the obligation to $X\ \mathbf{sat}_\rho^{\geqslant q}\ S'_M(t' - t'' - t_0 - t''' - \delta)$, which follows immediately from the hypothesis. For (7.4), we can use the proof rule for delay to reduce the proof obligation to $X\ \mathbf{sat}_\rho^{\geqslant q}\ S'_M(t' - t'' - t_0 - \delta)$, which again follows immediately from the hypothesis.

This completes the proof of condition 2. □

Condition 3: We must prove

$$PROT' \ \mathbf{sat}_{\rho}^{\geqslant p} \ S''_{m+1} \mid S'''_{m} \wedge G$$

The condition $S''_{m+1} \mid S'''_{m} \wedge G$ is equivalent to

$$\left. \begin{array}{l} \mathsf{internal}\, A \Rightarrow \\ \quad \begin{pmatrix} \mathsf{no}\ out\ \mathsf{offered}\ (t, t+T_m] \\ \wedge\ out.x\ \mathsf{live\ from}(t+T_m, t+T_{m+1}] \end{pmatrix} \end{array} \right| \begin{array}{l} \mathsf{internal}\, A \Rightarrow \\ \quad \begin{pmatrix} \mathsf{no}\ out\ \mathsf{offered}\ (t, t+T_m] \\ \wedge\ \mathsf{beyond}\ t+T_m \Rightarrow lm.x\ \mathsf{at}\ t+T_m \end{pmatrix} \\ \wedge\ in.x\ \mathsf{at}\ t \end{array}$$

which is the condition that the $m+1$th attempt at transmission is successful, given that the first m are unsuccessful and another attempt at transmission is made. We can use rule 7.1.16 to simplify this to $S''_{m+1} \mid G_m$ where

$$G_m \mathrel{\widehat{=}} \mathsf{no}\ out\ \mathsf{offered}\ (t, t+T_m] \wedge (\mathsf{beyond}\ t+T_m \Rightarrow lm.x\ \mathsf{at}\ t+T_m) \wedge in.x\ \mathsf{at}\ t$$

We will use the proof rule for parallel composition to reduce the proof obligation to showing $S \underset{conf}{\parallel\!\!\!+} R \ \mathbf{sat}_{\rho}^{\geqslant 1} \ S_{SR} \mid G_{SR}$ and $M \ \mathbf{sat}_{\rho}^{\geqslant p} \ S_M \mid G_M$ where

$$\begin{aligned} S_{SR} &\mathrel{\widehat{=}} \begin{pmatrix} \mathsf{internal}\ conf \wedge \mathsf{no}\ rm\ \mathsf{at}\ [t+T_m, t+T_m+t_0) \\ \wedge\ rm.x\ \mathsf{accessible\ from}\ t+T_m+t_0 \end{pmatrix} \Rightarrow out.x\ \mathsf{live\ from}\ t+T_{m+1} \\ G_{SR} &\mathrel{\widehat{=}} \mathsf{no}\ out\ \mathsf{offered}\ (t, t+T_m] \wedge (\mathsf{beyond}\ t+T_m \Rightarrow lm.x\ \mathsf{at}\ t+T_m) \wedge in.x\ \mathsf{at}\ t \\ S_M &\mathrel{\widehat{=}} lm.x\ \mathsf{at}\ t+T_m \Rightarrow \mathsf{no}\ rm\ \mathsf{at}\ [t+T_m, t+T_m+t_0) \wedge rm.x\ \mathsf{live\ from}\ t+T_m+t_0 \\ G_M &\mathrel{\widehat{=}} true \end{aligned}$$

$S_M \mid G_M$ is the condition that the input is correctly transmitted. $S_{SR} \mid G_{SR}$ is the condition that if an rm is offered by the medium at time $t+T_m+t_0$, then it becomes ready for output from $t+T_{m+1}$. We have the following proof obligation:

Lemma 7.4.2.3: If $\Sigma s \subseteq \{in, out, lm, rm, conf\}$ and $\sqsubseteq = \sqsubseteq_{SR} \underset{lm,rm}{\parallel\!\!\!+} \sqsubseteq_M$ then

$$\begin{pmatrix} S_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge G_{SR}(\tau, \sqsubseteq_{SR}, s) \\ \wedge\ S_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\}) \wedge G_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\}) \end{pmatrix} \Rightarrow S''_{m+1}(\tau, \sqsubseteq, s) \wedge G_m(\tau, \sqsubseteq, s)$$

and

$$G_m(\tau, \sqsubseteq, s) \Rightarrow G_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge G_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\})$$

♡

Proof of lemma: For the first obligation, assume

$$S_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge G_{SR}(\tau, \sqsubseteq_{SR}, s) \wedge S_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\}) \wedge G_M(\tau, \sqsubseteq_M, s \quad \{lm, rm\})$$

Then G_m follows immediately from G_{SR}. To prove S''_{m+1}, suppose internal A. Then from G_{SR} we have no out offered $(t, t+T_m]$. It remains to show $out.x$ live from$(t+T_m, t+T_{m+1}]$. If $\neg$ beyond $t+T_m$ then this is vacuously true, so suppose beyond $t+T_m$. Then from G_{SR} we have $lm.x$ at $t+T_m$ and so from S_M we have (no rm at $[t+T_m, t+T_m+t_0) \wedge rm.x$ live from $t+T_m+t_0)(\tau, \sqsubseteq_M, s \quad \{lm, rm\})$. We want to prove $(rm.x$ accessible from $t+T_m+t_0)(\tau, \sqsubseteq_{SR}, s)$: but this follows from corollary 5.3.2 by taking $c = rm.x$ and $C = \{rm\}$ and using the results of section 7.4.1. Hence the premises of S_{SR} are satisfied, so we have $out.x$ live from $t + T_{m+l}$. The second obligation follows trivially from the definitions. □

We now prove that $S \underset{conf}{\Vert} R$ satisfies its specification: $S \underset{conf}{\Vert} R \ \mathbf{sat}_\rho^{\geqslant l} \ S_{SR} \mid G_{SR}$. By rule 7.1.13 we can reduce the proof obligation to $S \underset{conf}{\Vert} R \ \mathbf{sat}_\rho \ G_{SR} \Rightarrow S_{SR}$. The predicate $G_{SR} \Rightarrow S_{SR}$ can be strengthened to $S'_{SR}(t+T_m)$, where

$$S'_{SR}(t') \mathrel{\widehat{=}} \begin{pmatrix} lm.x \text{ at } t' \wedge \text{internal } conf \\ \wedge \text{ no } rm \text{ at } [t', t'+t_0) \wedge rm.x \text{ accessible from } t'+t_0 \end{pmatrix} \Rightarrow out.x \text{ live from } t'+T$$

We will prove $S \underset{conf}{\Vert} R \ \mathbf{sat}_\rho \ S'_{SR}(t')$ for *all* t'. We reduce the proof obligation to $S \ \mathbf{sat}_\rho \ S_S$ and $R \ \mathbf{sat}_\rho \ S_R$, where

$$\begin{aligned} S_S &\mathrel{\widehat{=}} lm.x \text{ at } t' \Rightarrow conf_I \text{ live } [t'+\delta, t'+t_0+\delta] \\ S_R &\mathrel{\widehat{=}} rm.x \text{ at } t'+t_0 \Rightarrow \begin{pmatrix} conf_I \text{ live from } t'+t_0+\delta \\ \wedge \ conf_I \text{ at } t'+t_0+\delta \Rightarrow out.x \text{ live from } t'+T \end{pmatrix} \end{aligned}$$

We have the following proof obligation:

Lemma 7.4.2.4:

$$S_S(\tau, \sqsubseteq_S, s \quad \{in, lm, conf\}) \wedge S_R(\tau, \sqsubseteq_R, s \quad \{rm, conf, out\}) \Rightarrow S'_{SR}(\tau, \sqsubseteq_S \underset{conf}{\Vert} \sqsubseteq_R, s)$$

♡

Proof of lemma: Let $\sqsubseteq \mathrel{\widehat{=}} \sqsubseteq_S \underset{conf}{\Vert} \sqsubseteq_R$ and assume the premises. Suppose

$$lm.x \text{ at } t' \wedge \text{internal } conf \wedge \text{no } rm \text{ at } [t', t'+t_0) \wedge rm.x \text{ accessible from } t'+t_0$$

We must show $out.x$ live from $t'+T$. By the results of section 7.4.1 we know that R is ready to perform an rm from time $t'+t_0$, so $rm.x$ at $t'+t_0$. Hence from S_R we have $(conf_I$ live from $t'+t_0+\delta)(\tau, \sqsubseteq_R, s \quad \{rm, conf, out\})$ and from S_S we have $(conf_I$ live $[t'+\delta, t'+t_0+\delta])(\tau, \sqsubseteq_S, s \quad \{in, lm, conf\})$ so $conf_I$ at $t'+t_0+\delta$ since internal $conf$. Then from S_R we have $out.x$ live from $t'+T$, as desired. □

It is very easy to show that S and R satisfy their specifications using standard techniques.

We now prove that M satisfies its specification: $M \ \mathbf{sat}_\rho^{\geqslant p} \ S_M \mid G_M$. Since $G_M = true$, the proof obligation can be reduced to $M \ \mathbf{sat}_\rho^{\geqslant p} \ S'_M(t+T_m)$ where

$$S'_M(t') \mathrel{\widehat{=}} lm.x \text{ at } t' \Rightarrow \text{no } rm \text{ at } [t', t'+t_0) \wedge rm.x \text{ live from } t'+t_0$$

We use the proof rule for recursion to prove $\forall t' \quad M \ \mathbf{sat}_\rho^{\geqslant p} \ S'_M(t')$. Note that $STOP \ \mathbf{sat}_\rho \ S'_M(t')$ so the side condition of the proof rule is satisfied. Assume $\forall t' \quad X \ \mathbf{sat}_\rho^{\geqslant p} \ S'_M(t')$; we will show $\forall t' \quad lm?x \xrightarrow{t_0} (rm!x \longrightarrow X \ {}_p\sqcap_q \ WAIT\ \delta ; X) \ \mathbf{sat}_\rho^{\geqslant p} \ S'_M(t')$. Pick t'. We have two cases to consider:

- If the first lm occurs at t', then we can use the proof rule for prefixing to reduce the proof obligation to

$$rm!x \longrightarrow X \;{}_p\sqcap_q\; WAIT\ \delta\, ; X \ \mathsf{sat}_\rho^{\geqslant p}\ rm.x \text{ live from } 0$$

We can then use the proof rule for probabilistic choice to reduce the proof obligation to

$$rm!x \longrightarrow X \ \mathsf{sat}_\rho^{\geqslant 1}\ rm.x \text{ live from } 0 \qquad \text{and} \qquad WAIT\ \delta\, ; X \ \mathsf{sat}_\rho^{\geqslant 0}\ rm.x \text{ live from } 0$$

The first result follows from the rule for prefixing and the second result follows from rule 7.1.5.

- If the first lm occurs at time $t'' < t'$ then we can use the proof rule for prefixing to reduce the proof obligation to

$$rm!x \longrightarrow X \;{}_p\sqcap_q\; WAIT\ \delta\, ; X \ \mathsf{sat}_\rho^{\geqslant p}\ S'_M(t' - t'' - t_0)$$

We can then use the proof rule for probabilistic choice to reduce the proof obligation to

$$rm!x \longrightarrow X \ \ \mathsf{sat}_\rho^{\geqslant p}\ \ S'_M(t' - t'' - t_0) \tag{7.5}$$
$$WAIT\ \delta\, ; X \ \ \mathsf{sat}_\rho^{\geqslant p}\ \ S'_M(t' - t'' - t_0) \tag{7.6}$$

For (7.5), suppose the first rm occurs after a delay of t'''; then we can use the proof rule for prefixing to reduce the proof obligation to $X \ \mathsf{sat}_\rho^{\geqslant p}\ S'_M(t' - t'' - t_0 - t''' - \delta)$, which follows immediately from the hypothesis. For (7.6), we can use the proof rule for delay to reduce the proof obligation to $X \ \mathsf{sat}_\rho^{\geqslant p}\ S'_M(t' - t'' - t_0 - \delta)$, which again follows immediately from the hypothesis.

This completes the proof of condition 3. □

This completes the proof. □

7.4.3 Lessons learnt

We believe that we have learned a lot about doing proofs concerning probabilistic processes during the course of the above case study. Firstly, it's hard! One has to consider conditional specifications, which makes all our predicates more complicated than in unprobabilistic proofs. This factor also complicates our proof rules, as does the problem of sometimes having to reduce a proof obligation on a composite process to several proof obligations on the subcomponents. One also has to be very careful about quantification, because of the fact that universal quantification does not distribute through probabilistic specification; to get around this one has to be fairly explicit about when one is quantifying.

We believe that there are a number of ways that proofs involving probabilities can be made easier. Firstly, doing proofs about non-probabilistic aspects of the system can often help. In the example of a communications protocol, we began by proving a safety property that did not involve probabilities. During this proof we proved various results — particularly about the ordering of events — that were useful in the liveness proof.

It is also worth keeping the predicates involved as simple as possible. In particular, the right-hand sides of conditional specifications should be simplified wherever possible. For example, when using the proof rule for parallel composition to reduce an obligation of the form $P \parallel Q\ \mathbf{sat}_\rho^{\geqslant pq}\ S \mid G$, one normally seeks predicates S_P, G_P, S_Q and G_Q such that $P\ \mathbf{sat}_\rho^{\geqslant p}\ S_P \mid G_P$ and $Q\ \mathbf{sat}_\rho^{\geqslant q}\ S_Q \mid G_Q$, and such that

$$S_P(\tau, \sqsubseteq_P, s) \wedge G_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \wedge G_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \parallel \sqsubseteq_Q, s)$$

and

$$G(\tau, \sqsubseteq_P \parallel \sqsubseteq_Q, s) \Leftrightarrow G_P(\tau, \sqsubseteq_P, s) \wedge G_Q(\tau, \sqsubseteq_Q, s)$$

It is the last condition that seems to cause the problems. If G is a complicated predicate it is often not possible to find suitable predicates G_P and G_Q. This is because one can often not prove results about the behaviour of P simply from knowledge about the behaviour of $P \parallel Q$: one needs to know about the behaviour of Q as well. Fortunately the right hand side of conditional specifications can normally be simplified, for example via rules 7.1.16 and 7.1.17.

Parameterization of predicates is a useful technique: this allows a result to be deduced as a particular instance of a more general result. For example, in proving condition 3 in section 7.4.2 we had to show that the medium satisfied the condition that if an input was received at time $t + T_m$ then with probability p it was offered for output after a delay of length t_0:

$$M\ \mathbf{sat}_\rho^{\geqslant p}\ lm.x \text{ at } t + T_m \Rightarrow rm.x \text{ live from } t + T_m + t_0$$

where

$$M \mathrel{\widehat{=}} lm?x \xrightarrow{t_0} (rm!x \longrightarrow M\ {}_p\sqcap_q\ WAIT\ \delta\,;M)$$

We deduced this from the more general result

$$\forall t' \quad M\ \mathbf{sat}_\rho^{\geqslant p}\ S_M(t') \qquad \text{where} \qquad S_M(t') \mathrel{\widehat{=}} lm.x \text{ at } t' \Rightarrow rm.x \text{ live from } t' + t_0$$

Using the proof rule for recursion we assumed $\forall t' \quad M\ \mathbf{sat}_\rho^{\geqslant p}\ S_M(t')$ and sought to prove

$$\forall t' \quad lm?x \xrightarrow{t_0} (rm!x \longrightarrow M\ {}_p\sqcap_q\ WAIT\ \delta\,;M)\ \mathbf{sat}_\rho^{\geqslant p}\ S_M(t')$$

To do this we had to consider the case where the $lm.x$ resulted from a recursive call to M: if this recursive call started at time t'' then we used $S_M(t' - t'')$ to deduce our result. The point is that we could not have done this without the generalization of the result we were seeking to prove.

Chapter 8

Conclusions

In this thesis we have produced two languages based upon Timed CSP which can be used for arguing about priorities and probabilities in timed, communicating processes.

We have produced a language where many of the CSP operators have been refined so as to introduce a notion of priority. We have given a semantics for this language which models a process by the set of behaviours that it can perform, where the representation of a behaviour includes a record of the priorities given to different actions.

We have then extended the language to include a probabilistic choice operator. This has allowed us to present a semantics which models the probabilities of different behaviours occurring.

We presented a proof system for proving that a prioritized process meets its specification. We have also presented abstraction theorems from the Probabilistic and Deterministic Models to the (unprobabilistic) Prioritized Model, and have shown that a non-probabilistic specification on a probabilistic or deterministic process can be proved by showing that the corresponding prioritized process meets the same specification.

We have presented a specification language that allows one to make statements about when a process should perform events, when it should offer events, and what priorities different actions should have. The language is structured so as to make our specifications as readable as possible. It also has the advantage that the syntax is fairly independent of our semantic model — indeed much of the syntax is the same as the specification language in [DRRS93] — which means that most of our specifications can be interpreted in other models.

We have presented a complete set of proof rules for proving that a prioritized process meets its specification. These allow a proof obligation on a composite process to be reduced to proof obligations on its subcomponents. We have illustrated the proof system with several examples.

We have investigated how the Prioritized Model relates to the Timed Failures Model, and used this so show how results about prioritized processes can be proved by arguing in the Failures Model. We described which failures could have resulted from a particular prioritized behaviour, and used this to give an abstraction mapping from the Prioritized Model to the Timed Failures Model. We derived a proof rule that allows us to prove that a BTCSP process satisfies some specification if its unprioritized abstraction satisfies a corresponding specification. The Timed Failures Model is easier to reason with, and so this method should simplify many of our proofs. We then showed that our specification language was designed in such a way that the forms of many of our specifications were unchanged when translated into the Failures Model. This method was illustrated by an example where we implemented a BTCSP specification by firstly finding a TCSP process that satisfied nearly all of the conjuncts of the specification, and then examining which of the BTCSP refinements satisfied the rest of the specification.

Finally, we presented a proof system that can be used for proving that a process meets a probabilistic specification. Proofs of probabilistic processes are considerably harder than in the unprobabilistic case. We have described various difficulties that arise when one has to consider probabilities, and have shown how these can be overcome. We have illustrated the proof system by using it to analyse the performance of a communications protocol.

We hope that the work presented in this thesis will make it easier to reason formally about priorities and probabilities in timed communicating processes.

In this final chapter we make some comparisons with related work, and give some pointers to future work using the models presented in this thesis.

8.1 Related work

In this section we discuss other models of concurrency that include either probabilities or priorities.

8.1.1 Probabilistic models

The work nearest to our own is Karen Seidel's [Sei92]. She has produced a probabilistic model of untimed CSP. She defines a semantics for her model in terms of probability measures on the space of infinite traces. She writes $P\ A$ for the probability that process P performs a trace from set A. For example, the process $STOP$ is defined by

$$STOP\ A \mathrel{\hat{=}} \begin{cases} 1 & \text{if } \langle\tau\rangle^\omega \in A \\ 0 & \text{otherwise} \end{cases}$$

where $\langle\tau\rangle^\omega$ is the infinite sequence of invisible events τ.

Operators are defined as transformations on probability measures. The probabilistic choice operator ${}_p\sqcap$ chooses in favour of its first argument with probability p and in favour of its second argument with probability $1 - p$. It has a semantic definition given by

$$P\ {}_p\sqcap\ Q\ A \mathrel{\hat{=}} p.\ P\ A + (1 - p).\ Q\ A$$

The probability of $P\ {}_p\sqcap\ Q$ performing a trace from A is the probability that P is chosen times the probability that P performs a trace from A, plus the probability that Q is chosen times the probability that Q performs a trace from A.

The prefixing operator is defined by

$$a \longrightarrow P\ A \mathrel{\hat{=}}\ P\ (\mathit{prefix}_a^{-1}(A)) \qquad \text{where } \forall\, u \quad \mathit{prefix}_a(u) = \langle a\rangle\ \ u$$

The process $a \longrightarrow P$ can perform a trace from A if P can perform a trace from $\mathit{prefix}_a^{-1} A$.

Parallel composition is defined by

$$P \parallel Q \quad A \mathrel{\widehat{=}} (\ P \ \times \ Q\ )(par^{-1}(A))$$
$$\text{where } \forall\, u, v \quad par(u,v) = \begin{cases} u & \text{if } u = v \\ (u \quad n) \quad \langle \tau \rangle^{\omega} & \text{if } u \quad n = v \quad n \wedge u_n \neq v_n \end{cases}$$

where $P \ \times \ Q$ is a product measure. $P \parallel Q$ can perform a trace w from A if P and Q can perform traces u and v such that $u = v = w$, *or* u and v first differ in the nth position and w consists of their common prefix followed by τs.

However, she is unable to give a definition for external choice in this way, for much the same reasons that external choice caused us problems: it is not possible, for example, to give the probability of $a \longrightarrow STOP \quad b \longrightarrow STOP$ performing an a.

In order to give a semantics to external choice she defines conditional probability measures. For all processes P, sets of traces A, and traces y, the expression $(\!| P |\!)(A, y)$ represents the probability that P performs a trace from A *given that* the environment is willing to perform the trace y (and nothing else). For example, prefixing can be defined by

$$(\!| a \longrightarrow P |\!)(A, y) \mathrel{\widehat{=}} \begin{cases} (\!| P |\!)(prefix_a^{-1} A, y') & \text{if } y = \langle a \rangle \quad y' \\ 1 & \text{if } y_0 \neq a \wedge \langle \tau \rangle^{\omega} \in A \\ 0 & \text{otherwise} \end{cases}$$

Using this model, she defines an external choice operator. Informally, the process $P_S \quad Q$ behaves like P when offered a trace in S, and like Q when offered a trace not in S. The semantics for this process is given by

$$(\!| P_S \quad Q |\!)(A, y) \mathrel{\widehat{=}} \begin{cases} (\!| P |\!)(A, y) & \text{if } y \in S \\ (\!| Q |\!)(A, y) & \text{if } y \notin S \end{cases}$$

This is only defined in the case that for all finite traces t

$$y \in S \wedge z \notin S \wedge y, z \in A \Rightarrow (\!| P |\!)(A, y) = (\!| Q |\!)(A, z)$$
$$\text{where } A = \{t \quad u \mid \text{u is an infinite trace}\}$$

otherwise the probability of an action occurring could depend on what the environment offers at some time in the future. This definition effectively refines the external choice operator so as to make it deterministic.

Unfortunately, it is not possible to give a semantic definition for hiding in this way: for any trace y offered by the environment to the process $P \setminus X$, there is not a unique y' that is offered to P: when X is hidden, the process P should be able to perform *any* trace y' such that $y' \setminus X = y$.

It is interesting that — as in the language presented in this thesis — her languages are based upon deterministic subsets of CSP.

Most other probabilistic process algebras are based upon CCS [Mil89], with operational rather than denotational semantics. For example, Giacalone et al. [GJS90] have introduced a probabilistic version of SCCS, called PCCS. The nondeterministic process summation is replaced by a probabilistic counterpart: $\sum_{i \in I} [p_i] E_i$ (where $p_i \in (0, 1]$, $\sum p_i = 1$) is the process that offers a probabilistic choice between the processes P_i. If more than one process could be chosen, then they are chosen with relative probabilities p_i. If the choice is being made between two processes then this is written $[p]P + [q]Q$. Their work differs from the work described in this thesis by not differentiating between internal and external choice; they use the probabilistic choice operator for both. For example, our process $P\ {}_p\sqcap_q\ Q$ would be written $[p]\tau.P + [q]\tau.Q$, where the τ is an invisible action which can be thought of as representing the choice being made.

Van Glabbeek et al. [vGSST90] discuss *reactive*, *generative* and *stratified* models of probabilistic processes.

- They define a *reactive* model to be one where the environment may only offer one event at a time. If the process can perform the offered event then it makes an internal state transition according to some probability distribution. For example, the process $\frac{1}{3}a.P + \frac{2}{3}a.Q + b.R$ will, after an a, act like P with probability $\frac{1}{3}$ and like Q with probability $\frac{2}{3}$. They give an operational semantics for this language, writing $P \xhookrightarrow{\alpha[p]} P'$ to mean that P can perform the action α and with probability p become P'. For example, the probabilistic choice operator is given a semantics by

$$\sum_{i \in I}[p_i]\alpha_i.E_i \xhookrightarrow{\alpha_i[p_i/r_i]}_i E_i \qquad \text{where } r_i = \sum\{\!|p_j \mid j \in I \wedge \alpha_j = \alpha_i|\!\}$$

  Here r_i is the sum of the probabilities associated with all α_i-transitions; the probability of acting like E_i after performing α_i is therefore p_i/r_i. The subscript i on the arrow is to distinguish between two otherwise identical transformations; for example:

$$\frac{1}{2}a.nil + \frac{1}{2}a.nil \xhookrightarrow{a[1/2]}_1 nil \qquad \frac{1}{2}a.nil + \frac{1}{2}a.nil \xhookrightarrow{a[1/2]}_2 nil$$

- In a *generative* model, the environment can offer a choice between two or more events, and the process makes the choice according to some probability distribution. For example, if the process $\frac{1}{6}a.P + \frac{1}{3}a.Q + \frac{1}{2}b.R$ is offered an a or a b then it chooses the a with probability $\frac{1}{2}$ and the b with probability $\frac{1}{2}$; if the a is chosen, then the process acts like P with probability $\frac{1}{3}$, and like Q with probability $\frac{2}{3}$. If it is offered just an a, then the a will be performed (with probability 1), and it will then act like P with probability $\frac{1}{3}$, and like Q with probability $\frac{2}{3}$. The give an operational semantics for this model, writing $P \xrightarrow{\alpha[p]} P'$ to mean that with probability p, P will perform an α and then act like P'. The rule for choice is

$$E_j \xrightarrow{\alpha[q]}_k E' \quad \Rightarrow \quad \sum_{i \in I}[p_i]E_i \xrightarrow{\alpha[p_j \cdot q]}_{j,k} E' \quad (j \in I)$$

  In order to give a rule for the restriction operator, they define a function ν_G such that $\nu_G(E, A)$ gives the probability of process E performing an event from the set A:

$$\nu_G(E, A) \mathrel{\widehat{=}} \sum\{\!|p_i \mid \exists\, \alpha, E_i \quad E \xrightarrow{\alpha[p_i]}_i E_i \wedge \alpha \in A|\!\}$$

  Restriction is then defined by

$$E \xrightarrow{\alpha[p]}_i E' \quad \Rightarrow \quad E \quad A \xrightarrow{\alpha[p/r]}_i E' \quad A \qquad (\alpha \in A, r = \nu_G(E, A))$$

- A *stratified* model allows closer control over probabilistic choices. Consider a process that should choose an a with probability $\frac{1}{3}$, and otherwise choose between b and c with equal probabilities. The obvious definition of this is $P \mathrel{\widehat{=}} \frac{1}{3}a + \frac{1}{3}b + \frac{1}{3}c$. Consider however the case when the c is unavailable; with this definition, the a and the b are each chosen with probability $\frac{1}{2}$, rather than the desired $\frac{1}{3}$ and $\frac{2}{3}$. The process we require is $Q \mathrel{\widehat{=}} \frac{1}{3}a + \frac{2}{3}(\frac{1}{2}b + \frac{1}{2}c)$. If the c is unavailable then the a is chosen with probability $\frac{1}{3}$, and the b with probability $\frac{2}{3}$. The stratified model allows probabilistic choices between arbitrary processes. However it has no mechanism for allowing the environment to make a choice between processes. The operational semantics is defined via two transition relations:
  - an *action transition relation*, written $P \xrightarrow{\alpha} Q$: this has the normal definition except there is no rule for summation;
  - a *probability transition relation*, written $P \overset{p}{\longmapsto} Q$, meaning that with probability p, P will act like Q.

  The rule for the choice operator is

  $$\sum_{i \in I}[p_i]E_i \overset{p_i}{\longmapsto}_i E_i$$

  For the restriction operator, they define a function ν_S such that $\nu_S(E, A)$ gives the sum of the probabilities associated with transitions from A.

  $$\nu_S(E, A) \mathrel{\widehat{=}} \begin{cases} 1 & \text{if } E \xrightarrow{\alpha}, \text{ for } \alpha \in A \\ 0 & \text{if } E \xrightarrow{\beta}, \text{ for } \beta \notin A \\ \sum\{\!|p_i \mid E \overset{p_i}{\longmapsto}_i E_i \wedge \nu_S(E_i, A) \neq 0|\!\} & \text{otherwise} \end{cases}$$

  Restriction is then defined by

  $$E \overset{p}{\longmapsto}_i E' \wedge \nu_S(E', A) \neq 0 \quad \Rightarrow \quad E \quad A \overset{p/r}{\longmapsto}_i E' \quad A \qquad \text{where } r = \nu_S(E, A)$$

  The clause $\nu_S(E', A) \neq 0$ prevents the process from making a probabilistic transition into a state from where it can make no further A-transitions.

For each model they use the operational semantics to define strong bisimulations. They then give abstraction mappings between the three models.

The model described in this thesis does not fit comfortably into any of these categories. We can model the two processes P and Q that the stratified model is designed to distinguish by

$$P \mathrel{\widehat{=}} a \;{}_{1/3}\sqcap_{2/3}\; (b \;{}_{1/2}\sqcap_{1/2}\; c) \qquad Q \mathrel{\widehat{=}} a \;{}_{1/3}\sqcap_{2/3}\; (b \;{}_{1/2} \quad {}_{1/2}\; c)$$

We can distinguish these processes because we have included separate operators for internal and external choice. However, unlike in the stratified model, we are able to describe processes that offer the environment a choice between actions.

Tofts [Tof90] uses a weighted version of SCCS [Mil83]. For example, he writes $mP + nQ$ $(m, n \in \quad)$ for the process which will perform m occurrences of P for every n occurrences of Q.

The advantage of using weights rather than probabilities is that it makes renormalization unnecessary. For example, the rule for restriction is

$$\frac{E \overset{w}{\longmapsto} E' \qquad does_A(E')}{E \quad A \overset{w}{\longmapsto} E' \quad A}$$

where $does_A(E')$ is true if E' can perform an event from A; it is defined by

$$\frac{E \xrightarrow{a} E'}{does_A(E)}\Big[\, a \in A \,\Big] \qquad\qquad \frac{E \overset{w}{\longmapsto} E' \qquad does_A(E')}{does_A(E)}$$

Jou and Smolka [JS90] discuss various notions of process equivalence for probabilistic processes. They lift the notions of trace [Hoa85], maximal trace [BW82], failures [BHR84], maximal failures, ready [OH83] and bisimulation [Mil89] equivalence to the probabilistic case. They show that, unlike in the unprobabilistic case, maximal trace equivalence is no stronger than trace equivalence, and maximal failure equivalence is no stronger than failure equivalence. They also show that trace equivalence and failures equivalence are not congruences. For example, consider the processes

$$P \;\widehat{=}\; a.\left(\frac{1}{3}a + \frac{1}{3}b + \frac{1}{3}c\right)$$
$$Q \;\widehat{=}\; \frac{1}{2}a.\left(\frac{1}{3}a + \frac{1}{2}b + \frac{1}{6}c\right) + \frac{1}{2}a.\left(\frac{1}{3}a + \frac{1}{6}b + \frac{1}{2}c\right)$$

P and Q are trace and failures equivalent, but $P \quad \{a,c\}$ and $Q \quad \{a,c\}$ are not since $P \quad \{a,c\}$ will perform the trace $\langle a,c\rangle$ with probability $1/2$ while $Q \quad \{a,c\}$ will perform this trace with probability $7/15$. This result explains why we were not able to give a compositional denotational semantics based upon failures for our language — the result can be adapted to any language with a probabilistic external choice operator. Jou and Smolka then go on to give a complete axiomatization of probabilistic bisimulation.

Christoff [Chr90] defines three equivalences based on testing. He defines a *test* to be an unprobabilistic transition system that offers events to a process; he defines a *sequential test* to be a test that offers at most one event at a time. Note that the probability of a process performing a particular trace depends upon the test that provides its environment. He defines three equivalences as follows.

Probabilistic trace equivalence: He writes $s =_{tr} s'$ if, for all traces σ and all sequential tests t, processes s and s' have the same probability of performing trace σ in environment t. Note that the restriction to *sequential* tests means that this is equivalent to the reactive model of [vGSST90].

Weak probabilistic test equivalence: He writes $s =_{wte} s'$ if, for all tests, after performing trace σ the processes s and s' have the same probability of deadlock.

Strong probabilistic test equivalence: He writes $s =_{ste} s'$ if, for all tests, s and s' have the same probability of performing any trace σ.

He then defines three denotational functions. He defines an offering o to be a sequence of sets of events: intuitively these are the sets of events offered to a process at each stage. He defines a function μ such that $\mu(s, o\ \ L, \sigma\ \ a)$ is the probability that process s, given that it performs trace σ when offered o, goes on to perform an a when offered L. He uses this to define three denotational models, each representing a process by a probability function.

Probabilistic trace result systems: He defines the probability function μ_{tr} by

$$\mu_{tr}(\sigma) \mathrel{\hat{=}} \mu(s, Sets(\sigma), \sigma)$$

where for example $Sets(a, b, c) = \langle\{a\}, \{b\}, \{c\}\rangle$. Intuitively this gives the probability of performing the last element of σ, given that the environment offers this but nothing else, and given that the process has already performed the rest of σ.

Weak probabilistic test result systems: He defines the probability function μ_{wte} by

$$\mu_{wte}(o\ \ L, \sigma) \mathrel{\hat{=}} \sum\{\!|\mu(s, o\ \ L, \sigma\ \ a) \mid a \in L|\!\}$$

Intuitively this is the probability of not deadlocking when offered L after performing trace σ when offered o.

Strong probabilistic test result systems: He defines the probability function μ_{ste} by

$$\mu_{ste}(o, \sigma) \mathrel{\hat{=}} \mu(s, o, \sigma)$$

Intuitively this is the probability of performing trace σ when offered o.

For each denotational model and corresponding testing equivalence, he shows that two processes are equivalent in the denotational model if and only if they are equivalent under the testing equivalence.

Hans Hansson [Han91] has produced a discretely timed probabilistic process algebra based upon CCS, called TPCCS. Processes in his language alternate between probabilistic states (denoted by P, P', etc) and action states (denoted by N, N', etc). In action states, the process offers the environment a choice between a number of different actions; after performing an action, the process evolves into a probabilistic state; the environment is only allowed to offer one action at a time, so this is a reactive model in the terminology of [vGSST90]. In probabilistic states, processes evolve into action states according to some probability distribution. Like us, he differentiates between external and probabilistic choice, writing $\sum_{i\in I} \alpha_i P_i$ for an external choice and $\sum_{i\in I}[p_i]N_i$ for a probabilistic choice.

He begins by discussing an untimed language. Writing E_P for the probabilistic states and E_N for the action states he defines two relations $\hookrightarrow$: $E_N \times (Act \cup \{\tau\}) \times E_P$ and $\longmapsto$: $E_P \times [0, 1] \times E_N$, such that $N \overset{\alpha}{\hookrightarrow} P$ means that N can perform an α and become P, and $P \overset{p}{\longmapsto} N$ means that P can act like N with probability p. For example, the probabilistic choice operator is given a semantics by

$$\sum_{i\in I}[p_i]N_i \overset{p}{\longmapsto} N_i \qquad \text{where } p = \sum\{\!|p_j \mid N_j \equiv N_i \wedge j \in I|\!\}$$

Because his probabilistic choice operator is, like ours, internal rather than external, the definition for restriction is very straightforward:

$$\frac{P \overset{p}{\longmapsto} N}{P \setminus a \overset{p}{\longmapsto} N \setminus a} \qquad\qquad \frac{N \overset{\beta}{\hookrightarrow} P}{N \setminus a \overset{\beta}{\hookrightarrow} P \setminus a}\left[\beta, \bar{\beta} \neq a\right]$$

He then extends this language to a timed language by adding a special actiou χ which represents the passage of one unit of time. For example, he has the rule

$$\sum_{i \in I} \alpha_i.P_i \overset{\chi}{\hookrightarrow} [1] \sum_{i \in I} \alpha_i.P_i$$

(The $[1]$ here is to maintain the alternation between action and probabilistic states.) He then defines a timeout operator by

$$\frac{N \overset{\alpha}{\hookrightarrow} P'}{N \quad P \overset{\alpha}{\hookrightarrow} P'}\left[\alpha \neq \chi\right] \qquad\qquad N \quad P \overset{\chi}{\hookrightarrow} P$$

If N can perform an action to become P' then $N \quad P$ can perform that action to become P'; alternatively the process can timeout by performing a χ, and then act like P.

Unfortunately, the semantics defined by this relation does not satisfy the maximal progress assumption: a process may perform a χ when it could alternatively have performed a τ. To overcome this he defines a new relation $\longrightarrow$: $E_N \times (Act \cup \{\tau, \chi\}) \times E_P$ by

$$\frac{N \overset{\alpha}{\hookrightarrow} P}{N \overset{\alpha}{\longrightarrow} P}\left[\alpha \neq \chi\right] \qquad\qquad \frac{N \overset{\chi}{\hookrightarrow} P \quad N \overset{\tau}{\not\hookrightarrow}}{N \overset{\chi}{\longrightarrow} P}$$

Now N can only perform a χ if it is unable to perform a τ.

He then defines a branching time temporal logic TPCTL, based upon CTL [CES83], which allows one to specify properties such as "after a request for service there is at least a 98% probability that the service will be carried out within 2 seconds". He describes an algorithm for checking whether a TPCCS process satisfies a TPCTL specification.

Fang et. al. [FZHS92] have produced a probabilistic version of PARTY [HSZFH92]. They define three transition relations:

- they write $P \overset{a}{\longrightarrow} Q$ to denote that P can perform an a to become Q;
- they write $P \overset{a}{\longrightarrow}\bullet$ to denote that P can perform an a and terminate;
- they write $P \overset{p}{\Longrightarrow} Q$ to denote that with probability p, P acts like Q.

The definitions of these are quite straightforward. For example

$$\sum_{i \in I} [p_i] P_i \overset{p_i}{\Longrightarrow}_i P_i$$

They specify that probabilistic choices take one unit of time to be resolved. They use this to define a process $\langle t \rangle$ that terminates after t time units by

$$\begin{aligned} \langle 1 \rangle &\mathrel{\hat=} [1]\tau \\ \langle t \rangle &\mathrel{\hat=} [1]\langle t-1 \rangle \qquad \text{for } t > 1 \end{aligned}$$

Larsen and Skou [LS92] have investigated compositional verification of probabilistic processes. They define a logic, Probabilistic Model Logic (PML) with syntax given by

$$F ::= \mathit{true} \mid F \wedge F \mid \neg F \mid \langle a \rangle_p F$$

Intuitively $\langle a \rangle_p F$ specifies that a process can perform an a, and then with probability at least p go into a state that satisfies F. They define a simple reactive probabilistic language and then attempt to produce a system for decomposing logical specifications with respect to the unary operators of the language: for each unary operator O they seek a specification transformer $\mathcal{W}_O$ such that for any specification S and process P

$$O(P) \models S \quad \text{if and only if} \quad P \models \mathcal{W}_O(S)$$

In other words they seek to find the weakest specification $\mathcal{W}_O(S)$ for a component P that implies that a specification S holds for a composite process $O(P)$. However they show that this is not always possible using PML, for much the same reasons that in chapter 7 we were not always able to reduce a probabilistic specification on a composite process to a *single* specification on the subcomponents. They therefore introduce Extended Probabilistic Logic with the following syntax:

$$F ::= \mathit{true} \mid F \wedge F \mid \neg F \mid [\langle a \rangle_{x_1} F_1, \ldots, \langle a \rangle_{x_n} F_n \text{ where } \varphi(x_1, \ldots, x_n)]$$

The final clause has the intuitive meaning that the process can perform an a, and then with probability x_i go into a state that satisfies F_i (for each i), where the x_is satisfy the formula $\varphi(x_1, \ldots, x_n)$. They then show that this extended logic does support decomposition.

Jonsson and Larsen [JL91] have studied refinement between probabilistic processes, and used this as a method of proving that processes meet specifications. They represent a specification as a probabilistic transition system where each transition is labelled with a *set* of probabilities. They then define a satisfaction relation between processes and relations with the intuitive meaning that P **sat** S if the probability that labels a transition of P must be a member of the set of probabilities that labels the corresponding transition of S. They define refinement between specifications by saying that S refines T (written $S \subseteq T$) if P **sat** T whenever P **sat** S. In the case where S can be considered a process (i.e. if transitions are labelled with a *single* probability) then S **sat** T precisely when $S \subseteq T$. They present a complete, although complex, method of verifying that a process meets a specification. They then define another relation on specifications: T *simulates* S if whenever S can do a probabilistic transition, T can do likewise (but not necessarily vice versa). They show that if T *simulates* S then $S \subseteq T$. The advantage of nsing simulation over the previous verification method is that it is easier to test.

8.1.2 Prioritized models

In this subsection we discuss other models of concurrency that include priorities. In [CH88], Cleaveland and Hennessy describe a process algebra that uses prioritized actions rather than having prioritizing operators. They write $\underline{a}$ for a prioritized version of the action a. They define the semantics of their language in two stages. In the first stage, they define a relation $\longrightarrow$ which gives the normal semantics of CCS, ignoring priorities. They then define a relation $\longrightarrow\!\!\triangleright$ which takes account of priorities by

1. if $p \xrightarrow{\underline{\alpha}} q$ then $p \xrightarrow{\underline{\alpha}}\!\!\triangleright\, q$;

2. if $p \xrightarrow{\alpha} q$ and there are no q' and β such that $p \xrightarrow{\underline{\beta}} q'$, then $p \xrightarrow{\alpha}\!\!\triangleright\, q$.

This allows unprioritized events to happen only if no prioritized event can be performed. Note that the prioritized event $\underline{\alpha}$ can synchronise only with the prioritized event $\underline{\bar{\alpha}}$ and so they avoid the problem of opposing priorities on either side of a parallel composition. A strong bisimulation $\sim_p$ is defined from this relation in the normal way. However this is not a congruence because it identifies processes that can intuitively be distinguished: for example, $\underline{a}.p + b.q \sim_p \underline{a}.p$ but $(\underline{a}.p + b.q) \setminus \underline{a} \not\sim_p (\underline{a}.p) \setminus \underline{a}$ (where $\setminus \underline{a}$ is the CCS restriction operator that prevents $\underline{a}$ from occurring) because the former process can perform a b whereas the latter cannot. They therefore define a new relation $\rightarrowtail\!\!\triangleright$ by

1. if $p \xrightarrow{\underline{a}} q$ then $p \overset{\underline{a}}{\rightarrowtail\!\!\triangleright} q$;

2. if $p \xrightarrow{a} q$ and there is no q' such that $p \xrightarrow{\underline{\tau}} q'$, then $p \overset{a}{\rightarrowtail\!\!\triangleright} q$.

As before prioritized events are not constrained, but now unprioritized events are pre-empted only by $\underline{\tau}$. This relation is used to define a strong bisimulation, which they show to be a congruence.

Baeten et al. [BBK85] have produced a prioritized version of ACP called ACP_θ. They assume the existence of a partial ordering $>$ such that $a > b$ if a has a higher priority than b. They define an auxiliary operator $\lhd$ by

P1 $a \lhd b = a$ if not $(b > a)$

P2 $a \lhd b = \delta$ if $b > a$

where δ denotes deadlock. $a \lhd b$ is equal to a unless b has a higher priority than a. They introduce a priority operator θ such that $\theta(x)$ gives the behaviour of x in the given context. They define ACP_θ by adding the axioms **P1**–**P6** and **TH1**–**TH3** to ACP:

P3 $x \lhd y.z = x \lhd y$

P4 $x \lhd (y + z) = (x \lhd y) \lhd z$

P5 $x.y \lhd z = (x \lhd z).y$

P6 $(x + y) \lhd z = x \lhd z + y \lhd z$

TH1 $\theta(a) = a$

TH2 $\theta(x.y) = \theta(x).\theta(y)$

TH3 $\theta(x + y) = \theta(x) \lhd y + \theta(y) \lhd x$

This means that in a context where a has a higher priority than b ($a > b$), we have

$$\theta(a + b) = \theta(a) \lhd b + \theta(b) \lhd a = a \lhd b + b \lhd a = a + \delta = a$$

so the a takes precedence over the b. The difference between this and other models is that priorities between actions cannot change: if $a > b$ in some state then $a > b$ in all states.

In [Cam89], Camilleri defines a version of CCS [Mil89] with a left biased choice operator, $\mathrel{+\!\!\rangle}$ (confusingly, his arrow points away from the prioritized process, contrary to our convention of having arrows pointing towards prioritized processes). He defines the *acceptances* of a process: t **acc** A if A is the set of complements of the events that t can perform. He then defines the semantics of his language in terms of a transition system where $\vdash_R t_0 \xrightarrow{\mu} t_0'$ denotes that if the process t_0 is placed in an environment that refuses to perform events from R, then it can perform the event μ and then act like t_0'. He defines the biased choice operator by

$$\frac{\vdash_R t_0 \xrightarrow{\mu} t_0'}{\vdash_R t_0 \mathrel{+\!\!\rangle} t_1 \xrightarrow{\mu} t_0'} \qquad\qquad \frac{\vdash_R t_1 \xrightarrow{\mu} t_1' \quad t_0 \ \mathbf{acc}\ A}{\vdash_R t_0 \mathrel{+\!\!\rangle} t_1 \xrightarrow{\mu} t_1'}\Big[A \subseteq R\Big]$$

The non-prioritized process can only perform an event if the environment refuses to synchronise with any of the events of the prioritized process. So for example

$$\begin{array}{ll} \vdash_R a.t_0 \mathrel{+\!\!\rangle} b.t_1 \xrightarrow{a} t_0 & \text{for any } R \\ \vdash_R a.t_0 \mathrel{+\!\!\rangle} b.t_1 \xrightarrow{b} t_1 & \text{if } \bar{a} \notin R \end{array}$$

This relation is used to define a strong bisimulation $\sim_p$, which turns out to be a congruence. The problems of Cleaveland and Hennessy, described above, do not arise because $\vdash_{\{\bar{a}\}} a.p \mathrel{+\!\!\rangle} b.q \xrightarrow{b} q$, whereas $a.p$ cannot perform a b, so $a.p \mathrel{+\!\!\rangle} b.q \not\sim_p a.q$. This model fails, however, to adequately model the case where processes with opposing priorities are placed in parallel: the process $(\alpha.P \mathrel{+\!\!\rangle} \beta.Q) \mid (\bar{\beta}.Q' \mathrel{+\!\!\rangle} \bar{\alpha}.P')$ deadlocks immediately despite the fact that both sides of the parallel composition are able to perform either an a or a b.

Smolka and Steffen [SS90] consider priority as an extreme form of probability. Their work is based upon the stratified model of PCCS described above, but extended so as to allow zero probabilities. For example, $1a.P + 0b.Q$ will perform a b with probability 0, which they equate with impossibility. However $(1a.P + 0b.Q)\ \ b$ can perform a b. Thus this is a sort of prioritized choice in that $1a.P + 0b.Q$ can only perform a b in a context where an a is unavailable. However, as in the stratified model, processes cannot give the environment a choice between events. They give an operational semantics to this language in the same way as for the stratified language. The rule for restriction is

$$E \xmapsto{p}_s E' \wedge \nu_\zeta(E', A) \neq \perp \quad \Rightarrow \quad E\ \ A \xmapsto{r}_s E'\ \ A \quad (r = \rho_\zeta(E, A, p)) \qquad (*)$$

Informally, $\nu_\zeta(E', A)$ gives the sum of the probabilities of transitions from E' labelled with events from A, where the empty sum is taken to be $\perp$; hence the clause $\nu_\zeta(E', A) \neq\perp$ is true if E' can do *some* A-transition (possibly with probability 0). ν_ζ is defined by

$$\nu_\zeta(E, A) \mathrel{\hat=} \begin{cases} 1 & \text{if } E \xrightarrow{\alpha} \text{ for } \alpha \in A \\ \perp & \text{if } E \xrightarrow{\beta} \text{ for } \beta \notin A \\ \sum\{\!|p_i \mid E \overset{p_i}{\longmapsto}_i E_i \wedge \nu_\zeta(E_i, A) \neq\perp|\!\} & \text{otherwise} \end{cases}$$

The term $\rho_\zeta(E, A, p)$ which gives the probability of the transition in $(*)$ is then defined by

$$\rho_\zeta(E, A, p) \mathrel{\hat=} \begin{cases} \perp & \text{if } \nu_\zeta(E, A) = \perp \\ \frac{1}{n} & \text{if } \nu_\zeta(E, A) = 0 \text{ where } n \mathrel{\hat=} \#\{i \mid E \overset{0}{\longmapsto}_i E_i \wedge \nu_\zeta(E_i, A) \neq\perp\} \\ p/\nu_\zeta(E, A) & \text{if } \nu_\zeta(E, A) > 0 \end{cases}$$

If no A-transitions are available for E then the right hand side of $(*)$ is never applied, so in this case ρ_ζ is defined to be $\perp$. If only 0 probability transitions are available then the transitions are (arbitrarily) given equal probabilities. Otherwise, the probabilities in the non-restricted case are divided by the normalization factor ν_ζ.

Tofts [Tof90] extends the calculus of relative frequency, described above, to allow infinite weights. For example, he writes $\omega P + 1Q$ for the process that performs P infinitely more often than Q, i.e. the process that has an absolute priority towards P. The semantic definitions for this language are the same as for the language without infinite weights.

The programming language **occam** is closely based upon CSP. Therefore, it is useful to formally relate the two languages, and to use our experience of building models for CSP to produce models for **occam**. Brian Scott [Sco92] is currently working on this, and in particular he is working on a prioritized model of **occam**, based upon the prioritized model in this thesis.

8.2 Future work

The languages and models described in this thesis have opened up many directions for future work. I would like to:

- undertake further case studies;
- refine the models so as to make them easier to use;
- extend them so as to make them more expressive; and
- develop a tool for aiding reasoning about probabilistic processes.

In this thesis I have developed a number of techniques and useful rules for arguing about prioritized and probabilistic processes; however, I do not believe that our armoury is yet as complete as it could be. In order to further develop the craft of proving specifications for prioritized and probabilistic processes, and to find where further inference rules are needed, it will be necessary to carry out more case studies. In particular, proofs of probabilistic systems seem to be very different from proofs for unprobabilistic systems, so I would like to concentrate on these. There are a number of candidates for possible case studies, such as probabilistic consensus protocols [AH90, Sei92], mutual exclusion [PZ86], self stabilization [Her90], and communications protocols such as the alternating bit protocol [PS88, DS92b].

When proving that a process satisfies a specification, we are often faced with a situation where the specifications for the process and its subcomponents are expressed in terms of the specification language. To show that the specifications for the subcomponents are enough to imply the specification for the composite process we expand tbe macro definitions for the specifications — so as to express them in terms of our semantic representations of behaviours — and then apply the relevant proof rule, arguiug at the level of the semantics. For example, if we want to show that the process $P\ {}^{A}\Vert^{B}\ Q$ **sat** a live from t, where $a \in A \setminus B$, we might try to reduce this to proving that P **sat** a live from t. We cau do this by expanding the specification a live from t to

$$\forall\, t' \in [t, \infty) \quad (\exists\, t'' \in [t, t'] \quad a \in s(t'')) \vee t' \quad \tau \vee s \uparrow t' \uplus (t', \{\!|a|\!\}) \sqsupset s \uparrow t'$$

In order to use the proof rule for parallel composition, noting that wc must have Q sat *true*, we have to show that

$$\begin{pmatrix} \forall\, t' \in [t, \infty) \quad (\exists\, t'' \in [t, t'] \quad a \in s(t'') \quad A) \vee t' \quad \tau \vee s \quad A \uparrow t' \uplus (t', \{\!|a|\!\}) \sqsupset_P s \quad A \uparrow t' \\ \wedge\ \mathit{true} \end{pmatrix} \Rightarrow$$
$$\forall\, t' \in [t, \infty) \quad (\exists\, t'' \in [t, t'] \quad a \in s(t'')) \vee t' \quad \tau \vee s \uparrow t' \uplus (t', \{\!|a|\!\}) \sqsupset s \uparrow t'$$

A simpler way to argue would be if we had a rule of the form

$$\frac{P \textbf{ sat } a \text{ live from } t}{P\ {}^{A}\Vert^{B}\ Q \textbf{ sat } a \text{ live from } t}\left[\, a \in A \setminus B \,\right]$$

Then this rule could be applied directly. This would make our proofs easier to carry out, and easier to read. Equally, it would be useful to have similar rules for the Probabilistic Model, such as

$$\frac{P \textbf{ sat}^{\geqslant p}\ a \text{ live from } t}{P\ {}^{A}\Vert^{B}\ Q \textbf{ sat}^{\geqslant p}\ a \text{ live from } t}\left[\, a \in A \setminus B \,\right]$$

It would be useful to produce a library of such derived proof rules that argue at the level of the specification language. Jim Davies and Steve Schneider are currently developing rules of this form for the Timed Failures Model; these rules could be adapted to the prioritized and probabilistic models, and rules particular to these models could be developed by pursuing further case studies.

The probabilistic language described in this thesis is based upon a prioritized language; however, it is normally the case that when studying a particular probabilistic process, the choice of priorities upon the operators is completely arbitrary. It would therefore be useful to consider a language that includes probabilities but not priorities. In order to do this we will have to find a way of modelling nondeterminism in a probabilistic setting. I believe that in order to do this we will have to represent a process as a *set* of probability fnnctions, one function for each way the nondeterministic choices can be made. As we will no longer have to model priorities, it may be possible to base our representation of behaviours upon timed failures. However, developing the semantic definitions is likely to be particularly difficult.

These changes will considerably complicate the semantic model, but may lead to a proof system that is easier to use. For example, recall how the probabilistic rule for parallel composition —

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p} \ S_P \\ Q \ \mathbf{sat}_{\rho}^{\geqslant q} \ S_Q \\ S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \Vert\!\!\!\!\!\!\;\;\Vert \sqsubseteq_Q, s) \end{array}}{P \Vert\!\!\!\!\!\!\;\;\Vert Q \ \mathbf{sat}_{\rho}^{\geqslant pq} \ S}$$

— is related to the corresponding rule in the unprobabilistic, Prioritized Model —

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho} \ S_P \\ Q \ \mathbf{sat}_{\rho} \ S_Q \\ S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \Vert\!\!\!\!\!\!\;\;\Vert \sqsubseteq_Q, s) \end{array}}{P \Vert\!\!\!\!\!\!\;\;\Vert Q \ \mathbf{sat}_{\rho} \ S}$$

Similarly, I believe that we should be able to adapt the rule for parallel composition in the Timed Failures Model —

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho} \ S_P \\ Q \ \mathbf{sat}_{\rho} \ S_Q \\ S_P(s, \aleph_P) \wedge S_Q(s, \aleph_Q) \Rightarrow S(s, \aleph_P \cup \aleph_Q) \end{array}}{P \parallel Q \ \mathbf{sat}_{\rho} \ S}$$

— to a corresponding rule for a probabilistic model based upon timed failures —

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p} \ S_P \\ Q \ \mathbf{sat}_{\rho}^{\geqslant q} \ S_Q \\ S_P(s, \aleph_P) \wedge S_Q(s, \aleph_Q) \Rightarrow S(s, \aleph_P \cup \aleph_Q) \end{array}}{P \parallel Q \ \mathbf{sat}_{\rho}^{\geqslant pq} \ S}$$

The models described in this thesis have been timed. However, many systems can be adequately described without including timing information. It would be useful to have untimed models of probabilistic behaviour since this will make reasoning about such systems easier.

In this thesis we have only dealt with processes that can choose probabilistically between a countable number of behaviours. In order to reason about processes that can probabilistically choose between an uncountable number of behaviours — for example, the process that will perform an a after a random amount of time between 0 and 2 seconds with a uniform probability density — it will be necessary to extend the semantic model.

Proofs using the proof system for the probabilistic model tend to be extremely complicated. It would therefore be useful to have a proof tool to assist in these proofs.

I believe that it would also be useful to have a tool based upon the notion of refinement between probabilistic processes. The Failures, Divergences Refinement Checker (FDR) is a tool developed at Oxford for automatically testing whether a CSP process meets its specification.

Formally it takes two untimed processes P (the specification) and Q (the implementation), and tests whether Q refines P (written $P \sqsubseteq Q$), in the sense that Q can only behave in ways that P can behave. This corresponds to Q being more deterministic than P, or formally that $\mathcal{F}_U\ P\ \supseteq \mathcal{F}_U\ Q$ and $\mathcal{D}_U\ P\ \supseteq \mathcal{D}_U\ Q$ where the functions $\mathcal{F}_U$ and $\mathcal{D}_U$ give the untimed failures and divergences of a process. I would like to extend FDR in order to model probabilities.

Since the tool is based upon an operational — rather than a denotational — semantics, I will have to develop an operational semantics for a probabilistic language. There will be no real need to include priorities in this language: we included priorities in the language in this thesis only in order to rid ourselves of nondeterminism, so that we could actually predict the probability of any behaviour in a given situation; with an operational semantics there is no need to do this: indeed, since our notion of refinement is that of one process being more deterministic than another, it is essential for us to include nondeterminism in our language. I therefore intend to base the syntax upon untimed CSP, extended with a probabilistic choice operator. Producing an operational semantics for this language should be straight forward, following, for example, the work of Hansson [Han91].

I will also need to formally define what it means for one probabilistic process to refine another. I believe the correct definition will be to make the refinement relation the smallest relation such that

- $P \sqcap Q \quad P$;
- $P \sqcap Q \quad Q$;
- $P \sqcap Q \quad P\ {}_p\sqcap_q\ Q$ for any probabilities p and q such that $p + q = 1$;
- all the CSP operators are monotonic with respect to .

Looking at this another way, we will have $P \quad Q$ if there is some way of replacing some of the nondeterministic choices of P with probabilistic choices so that it is indistinguishable from Q: i.e. after any trace they have the same probability distributions on refusals.

This definition of refinement will, I believe, prove useful in allowing us to write specifications as probabilistic processes. For example, modelling the passage of time by the visible event *tock*, we can test whether a process performs the event a within 2 seconds with a probability of 99%, by testing whether it refines the process

$$\begin{array}{l}(a \longrightarrow CHAOS \sqcap tock \longrightarrow a \longrightarrow CHAOS)\\ {}_{.99}\sqcap_{.01}\\ CHAOS\end{array}$$

where $CHAOS$ is the most nondeterministic process. This specification says that with probability 99% an a must be performed after at most one *tock*.

Appendix A

Summary of Semantic Definitions

A.1 Subsidiary functions

$$\text{fillout}\, f\ (\tau, \sqsubseteq, s) \ \widehat{=}\ \begin{cases} f(\tau, \sqsubseteq, s) & \text{if } (\tau, \sqsubseteq, s) \in \text{dom} f \\ 0 & \text{if } (\tau, \sqsubseteq, s) \notin \text{dom} f \end{cases}$$

$$\begin{aligned}
\uparrow_{\sqsubseteq_P, \sqsubseteq_Q} w \ &\widehat{=}\ \sqcup_{\sqsubseteq_P} \{ w'_P \in \text{items} \sqsubseteq_P \mid w'_P \subseteq w \wedge w - w'_P \in \text{items} \sqsubseteq_Q \} \\
&\qquad \text{if } \exists\, w_P \in \text{items} \sqsubseteq_P, w_Q \in \text{items} \sqsubseteq_Q \quad w = w_P \uplus w_Q \\
\downarrow_{\sqsubseteq_P, \sqsubseteq_Q} w \ &\widehat{=}\ w - \uparrow_{\sqsubseteq_P, \sqsubseteq_Q} w \\
&\qquad \text{if } \exists\, w_P \in \text{items} \sqsubseteq_P, w_Q \in \text{items} \sqsubseteq_Q \quad w = w_P \uplus w_Q \\
\Uparrow^g_{\sqsubseteq} w \ &\widehat{=}\ \sqcup_{\sqsubseteq} \{ w' \in \text{items} \sqsubseteq \mid g w' = w \}. \qquad \text{if } \exists\, w' \in \text{items} \sqsubseteq \quad g w' = w \\
M(X, P)\rho \ &\widehat{=}\ \lambda\, Y \quad \mathcal{F}_{PBT}\ P\ \rho[Y/X] \\
W_\delta \ &\widehat{=}\ \lambda\, Y \quad \mathcal{F}_{PBT}\ \mathit{WAIT}\ \delta\, ;\, X\ \rho[Y/X] \\
M_\delta(X, P)\rho \ &\widehat{=}\ M(X, P)\rho \circ W_\delta \\
M(\underline{X}, \underline{P})\rho \ &\widehat{=}\ \lambda\, \underline{Y} \quad \mathcal{F}_{PBT}\ \underline{P}\ \rho[Y_i / X_i \mid i \in I]
\end{aligned}$$

A.2 Operations on offer relations

$$\begin{aligned}
v(\sqsubseteq_P \between \sqsubseteq_Q)w \ &\Leftrightarrow\ v \sqsubseteq_P w \\
&\qquad \vee\ w \in \text{items} \sqsubseteq_P \wedge \Sigma w \neq \{\!|\,|\!\} \wedge v \in \text{items} \sqsubseteq_Q \setminus \text{items} \sqsubseteq_P \\
&\qquad \vee\ v, w \notin \text{items} \sqsubseteq_P \wedge v \sqsubseteq_Q w \\
v(\sqsubseteq_P {}^X\!\!\Vert^Y \sqsubseteq_Q)w \ &\Leftrightarrow\ (v \quad X \sqsubset_P w \quad X \vee v \quad X = w \quad X \wedge v \quad Y \sqsubseteq_Q w \quad Y) \\
&\qquad \wedge\ v \quad X, w \quad X \in \text{items} \sqsubseteq_P \wedge v \quad Y, w \quad Y \in \text{items} \sqsubseteq_Q \\
&\qquad \wedge\ \Sigma v, \Sigma w \subseteq X \cup Y
\end{aligned}$$

$$
\begin{aligned}
v(\sqsubseteq_P \longleftarrow \sqsubseteq_Q)w \;\Leftrightarrow\; & \exists\, v'_P \in \text{items}\sqsubseteq_P, v'_Q \in \text{items}\sqsubseteq_Q \quad v = v'_P \uplus v'_Q \\
& \wedge \exists\, w'_P \in \text{items}\sqsubseteq_P, w'_Q \in \text{items}\sqsubseteq_Q \quad w = w'_P \uplus w'_Q \\
& \wedge v_P \sqsubset_P w_P \vee v_P = w_P \wedge v_Q \sqsubseteq_Q w_Q \\
& \text{where} \quad v_P = \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} v \qquad v_Q = \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} v \\
& \qquad\quad\;\; w_P = \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} w \qquad w_Q = \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} w
\end{aligned}
$$

$$v(\sqsubseteq \setminus X)w \;\Leftrightarrow\; \exists\, v', w' \in \text{items}\sqsubseteq \quad v' \setminus X = v \wedge w' \setminus X = w \wedge \Uparrow_{\sqsubseteq}^{\setminus X} v \sqsubseteq \Uparrow_{\sqsubseteq}^{\setminus X} w$$

$$v(g \odot \sqsubseteq)w \;\Leftrightarrow\; \exists\, v' \in \text{items}\sqsubseteq \quad gv' = v \wedge \exists\, w' \in \text{items}\sqsubseteq \quad gw' = w \wedge \Uparrow_{\sqsubseteq}^{g} v \sqsubseteq \Uparrow_{\sqsubseteq}^{g} w$$

$$\sqsubseteq \oplus a \;\widehat{=}\; \sqsubseteq \longrightarrow I\sqsubseteq \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle$$

A.3 Semantic definitions

Let $A_P \mathrel{\widehat{=}} \mathcal{A}_{PBT}\ P\ \rho$, $A_Q \mathrel{\widehat{=}} \mathcal{A}_{PBT}\ Q\ \rho$, $f_P \mathrel{\widehat{=}} \mathcal{P}_{PBT}\ P\ \rho$, $f_Q \mathrel{\widehat{=}} \mathcal{P}_{PBT}\ Q\ \rho$.

$\mathcal{A}_{PBT}\ STOP\ \rho \mathrel{\widehat{=}} \{(\tau, [0, \tau] \otimes \langle \{\!|\,|\!\} \rangle, \{\!|\,|\!\})\}$

$\mathcal{P}_{PBT}\ STOP\ \rho \mathrel{\widehat{=}} \text{fillout}\{(\tau, [0, \tau] \otimes \langle \{\!|\,|\!\} \rangle, \{\!|\,|\!\}) \mapsto 1\}$

$\mathcal{A}_{PBT}\ WAIT\ t\ \rho \mathrel{\widehat{=}}$
$\quad \{(\tau, [0, \tau] \otimes \langle \{\!|\,|\!\} \rangle, \{\!|\,|\!\}) \mid \tau < t\}$
$\quad \cup\{(\tau, [0, t) \otimes \langle \{\!|\,|\!\} \rangle \quad [t, \tau] \otimes \langle \{\!|\quad|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \mid \tau \quad t\}$
$\quad \cup\{(\tau, [0, t) \otimes \langle \{\!|\,|\!\} \rangle \quad [t, t'] \otimes \langle \{\!|\quad|\!\}, \{\!|\,|\!\} \rangle \quad (t', \tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec(t', \quad)\succ) \mid t \quad t' \quad \tau\}$

$\mathcal{P}_{PBT}\ WAIT\ t\ \rho \mathrel{\widehat{=}}$
$\quad \text{fillout}\,(\{(\tau, [0, \tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec\succ) \mapsto 1 \mid \tau < t\}$
$\qquad\quad \cup\{(\tau, [0, t) \otimes \langle \{\!|\,|\!\} \rangle \quad [t, \tau] \otimes \langle \{\!|\quad|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \mapsto 1 \mid \tau \quad t\}$
$\qquad\quad \cup\{(\tau, [0, t) \otimes \langle \{\!|\,|\!\} \rangle \quad [t, t'] \otimes \langle \{\!|\quad|\!\}, \{\!|\,|\!\} \rangle \quad (t', \tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec(t', \quad)\succ) \mapsto 1 \mid t \quad t' \quad \tau\})$

$\mathcal{F}_{PBT}\ X\ \rho \mathrel{\widehat{=}} \rho\ X$

$\mathcal{A}_{PBT}\ a \xrightarrow{0} P\ \rho \mathrel{\widehat{=}}$
$\quad \{(\tau, [0, \tau] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle, \prec\succ)\}$
$\quad \cup\{\{(\tau, [0, t] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle \quad \sqsubseteq_P + t, (t, a) \quad s_P + t) \mid$
$\qquad\quad (\tau - t, 0 \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq_P, s_P) \in A_P \wedge \tau \quad t\}$

$\mathcal{P}_{PBT}\ a \xrightarrow{0} P\ \rho \mathrel{\widehat{=}}$
$\quad \text{fillout}\,(\{(\tau, [0, \tau] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \mapsto 1\}$
$\qquad\quad \cup\{(\tau, [0, t] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle \quad \sqsubseteq_P + t, (t, a) \quad s + t) \mapsto f_P(\tau - t, 0 \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq_P, s) \mid$
$\qquad\qquad \tau \quad t\})$

$$
\begin{aligned}
&\mathcal{A}_{PBT}\ P \quad Q\ \rho \mathrel{\widehat{=}}\\
&\quad \{(\tau, \sqsubseteq_P, s_P) \mid \forall t \quad s_P \uparrow t \in \text{items} \sqsubseteq_P\\
&\qquad\qquad \wedge \exists \sqsubseteq'_P \quad \sqsubseteq'_P \setminus \quad = \sqsubseteq_P \wedge (\tau, \sqsubseteq'_P, \Uparrow^{-\backslash\checkmark}_{\sqsubseteq'_P} s_P) \in A_P \wedge \quad \notin \Sigma(\Uparrow^{-\backslash\checkmark}_{\sqsubseteq'_P} s_P)\}\\
&\quad \cup\{(\tau, \sqsubseteq_P \quad (t,\tau] \otimes \langle\{\!\mid\!\}\rangle . s_P) \mid\\
&\qquad t \quad \tau < t + \delta \wedge \forall t' \quad s_P \uparrow t' \in \text{items} \sqsubseteq_P\\
&\qquad \wedge \exists \sqsubseteq'_P \quad \sqsubseteq'_P \setminus \quad = \sqsubseteq_P \wedge (t, \sqsubseteq'_P, \Uparrow^{-\backslash\checkmark}_{\sqsubseteq'_P} s_P) \in A_P \wedge \mathit{begin}((\Uparrow^{-\backslash\checkmark}_{\sqsubseteq'_P} s_P) \quad) = t\}\\
&\quad \cup\{(\tau, \sqsubseteq_P \quad (t, t+\delta) \otimes \langle\{\!\mid\!\}\rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad s_Q + t + \delta) \mid\\
&\qquad t \quad \tau - \delta \wedge \forall t' \quad s_P \uparrow t' \in \text{items} \sqsubseteq_P\\
&\qquad \wedge \exists \sqsubseteq'_P \quad \sqsubseteq'_P \setminus \quad = \sqsubseteq_P \wedge (t, \sqsubseteq'_P, \Uparrow^{-\backslash\checkmark}_{\sqsubseteq'_P} s_P) \in A_P \wedge \mathit{begin}((\Uparrow^{-\backslash\checkmark}_{\sqsubseteq'_P} s_P) \quad) = t\\
&\qquad \wedge (\tau - (t+\delta), \sqsubseteq_Q, s_Q) \in A_Q\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{P}_{PBT}\ P \quad Q\ \rho \mathrel{\widehat{=}}\\
&\quad \text{fillout}\left\{
\begin{array}{l}
(\tau, \sqsubseteq, s) \mapsto\\
\quad \sum\{\!\mid f_P(\tau, \sqsubseteq_P, \Uparrow^{-\backslash\checkmark}_{\sqsubseteq_P} s) \mid \sqsubseteq_P \setminus \quad = \sqsubseteq \wedge \quad \notin \Sigma(\Uparrow^{-\backslash\checkmark}_{\sqsubseteq_P} s) \mid\!\}\\
\quad + \sum\left\{\!\left| f_P(t, \sqsubseteq_P, \Uparrow^{-\backslash\checkmark}_{\sqsubseteq_P} s) \middle| \begin{array}{l} \sqsubseteq = \sqsubseteq_P \setminus \quad (t,\tau] \otimes \langle\{\!\mid\!\}\rangle \\ \wedge\ \mathit{begin}((\Uparrow^{-\backslash\checkmark}_{\sqsubseteq_P} s) \quad) = t \wedge t \quad \tau < t + \delta \end{array} \right|\!\right\}\\
\quad + \sum\left\{\!\left| \begin{array}{l} f_P(t, \sqsubseteq_P, \Uparrow^{-\backslash\checkmark}_{\sqsubseteq_P} (s \quad t)). f_Q(\tau - (t+\delta), \sqsubseteq_Q, s - t - \delta) \mid \\ \qquad \sqsubseteq = \sqsubseteq_P \setminus \quad (t, t+\delta) \otimes \langle\{\!\mid\!\}\rangle \quad \sqsubseteq_Q + t + \delta \\ \qquad \wedge\ \mathit{begin}((\Uparrow^{-\backslash\checkmark}_{\sqsubseteq_P} s) \quad) = t \wedge t \quad \tau - \delta \end{array} \right|\!\right\}\\
\qquad\quad \forall t \quad s \uparrow t \in \text{items} \sqsubseteq
\end{array}
\right\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{A}_{PBT}\ \mathit{WAIT}\ t\,;P\ \rho \mathrel{\widehat{=}}\\
&\quad \{(\tau, [0,\tau] \otimes \langle\{\!\mid\!\}\rangle, \{\!\mid\!\}) \mid t > \tau\} \cup \{(\tau, \sqsubseteq + t, s + t) \mid t \quad \tau \wedge (\tau - t, \sqsubseteq, s) \in A_P\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{P}_{PBT}\ \mathit{WAIT}\ t\,;P\ \rho \mathrel{\widehat{=}}\\
&\quad \text{fillout}(\{(\tau, [0,\tau] \otimes \langle\{\!\mid\!\}\rangle, \{\!\mid\!\}) \mapsto 1 \mid t > \tau\}\\
&\qquad\quad \cup\{(\tau, \sqsubseteq, s) \mapsto f_P(\tau - t, \sqsubseteq - t, s - t) \mid t \quad \tau \wedge s \quad t = \prec\succ \wedge \sqsubseteq \quad t = [0, t) \otimes \langle\{\!\mid\!\}\rangle\})
\end{aligned}
$$

$$\mathcal{A}_{PBT}\ P\ {}_p\sqcap_q\ Q\ \rho \mathrel{\widehat{=}} A_P \cup A_Q$$

$$\mathcal{P}_{PBT}\ P\ {}_p\sqcap_q\ Q\ \rho(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} p.f_P(\tau, \sqsubseteq, s) + q.f_Q(\tau, \sqsubseteq, s)$$

$$\mathcal{A}_{PBT} \quad {}_{i \in I}[p_i]P_i\ \rho \mathrel{\widehat{=}} \bigcup\{\mathcal{A}_{PBT}\ P_i\ \rho \mid i \in I\}$$

$$\mathcal{P}_{PBT} \quad {}_{i \in I}[p_i]P_i\ \rho(\tau, \sqsubseteq, s) \mathrel{\widehat{=}} \sum\{\!\mid p_i \times \mathcal{P}_{PBT}\ P_i\ \rho(\tau, \sqsubseteq, s) \mid i \in I \mid\!\}$$

$$
\begin{aligned}
&\mathcal{A}_{PBT}\ P \Box Q\ \rho \mathrel{\widehat{=}}\\
&\quad \{(\tau, \sqsubseteq_P \Box \sqsubseteq_Q, \prec\succ) \mid (\tau, \sqsubseteq_P, \prec\succ) \in A_P \wedge (\tau, \sqsubseteq_Q, \prec\succ) \in A_Q\}\\
&\quad \cup\{(\tau, \sqsubseteq_P \Box \sqsubseteq_Q, s) \mid s \neq \prec\succ \wedge \mathit{begin}\ s = t \wedge (t, \sqsubseteq_P, \prec\succ) \in A_P\}\\
&\qquad\qquad\qquad\qquad \wedge (\tau, \sqsubseteq_Q, s) \in A_Q \wedge s \uparrow t \not\sqsupset_P (t, \{\!\mid\!\})\}\\
&\quad \cup\{(\tau, \sqsubseteq_P \Box \sqsubseteq_Q, s) \mid\\
&\qquad s \neq \prec\succ \wedge \mathit{begin}\ s = t \wedge (\tau, \sqsubseteq_P, s) \in A_P \wedge (t, \sqsubseteq_Q, \prec\succ) \in A_Q\\
&\qquad \wedge (s \uparrow t \sqsupset_P (t, \{\!\mid\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_Q)\}
\end{aligned}
$$

$$\mathcal{P}_{PBT}\ P \Box Q\ \rho(\tau, \sqsubseteq, \prec\succ) \mathrel{\widehat{=}} \sum\{\!\mid f_P(\tau, \sqsubseteq_P, \prec\succ).f_Q(\tau, \sqsubseteq_Q, \prec\succ) \mid \sqsubseteq = \sqsubseteq_P \Box \sqsubseteq_Q \mid\!\}$$

$$\mathcal{P}_{PBT}\ P \sqcap Q\ \rho(\tau, \sqsubseteq, s) \mathrel{\widehat{=}}$$
$$\sum \Big\{\!\!\Big| f_P(t, \sqsubseteq_P, \prec\succ).f_Q(\tau, \sqsubseteq_Q, s) \mid \sqsubseteq\ = \ \sqsubseteq_P \sqcap \sqsubseteq_Q \wedge s \uparrow t \not\sqsupseteq_P (t, \{\!|\,|\!\}) \Big|\!\!\Big\}$$
$$+ \sum \left\{\!\!\left| \begin{array}{l} f_P(\tau, \sqsubseteq_P, s).f_Q(t, \sqsubseteq_Q, \prec\succ) \mid \\ \quad \sqsubseteq\ = \ \sqsubseteq_P \sqcap \sqsubseteq_Q \wedge (s \uparrow t \sqsupseteq_P (t, \{\!|\,|\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_Q) \end{array} \right|\!\!\right\}$$
$$\text{if } s \neq \prec\succ\ \wedge\ \mathit{begin}\ s = t$$

$$\mathcal{A}_{PBT}\ c?a : A \xrightarrow{0} P_a\ \rho \mathrel{\widehat{=}}$$
$$\{(\tau, [0, \tau] \otimes \langle \{\!|c?a|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \mid \tau \in \mathit{TIME}\}$$
$$\cup \{(\tau, [0, t] \otimes \langle \{\!|c?a|\!\}, \{\!|\,|\!\} \rangle \quad \sqsubseteq + t, (t, c?\hat{a}) \quad s + t) \mid$$
$$\hat{a} \in A \wedge t \quad \tau \wedge (\tau - t, \{0\} \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq, s) \in \mathcal{A}_{PBT}\ P_{\hat{a}}\ \rho\}$$

$$\mathcal{P}_{PBT}\ c?a : A \xrightarrow{0} P_a\ \rho \mathrel{\widehat{=}}$$
$$\text{fillout}(\{(\tau, [0, \tau] \otimes \langle \{\!|c?a|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \mapsto 1 \mid \tau \in \mathit{TIME}\}$$
$$\cup \{(\tau, [0, t] \otimes \langle \{\!|c?a|\!\}, \{\!|\,|\!\} \rangle \quad \sqsubseteq + t, (t, c?\hat{a}) \quad s + t)$$
$$\mapsto \mathcal{P}_{PBT}\ P_{\hat{a}}\ \rho(\tau - t, \{0\} \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq, s) \mid$$
$$\hat{a} \in A \wedge t \quad \tau\})$$

$$\mathcal{A}_{PBT}\ P\ {}^X\!\|^Y\ Q\ \rho \mathrel{\widehat{=}}$$
$$\{(\tau, \sqsubseteq_P {}^X\!\|^Y \sqsubseteq_Q, s) \mid (\tau, \sqsubseteq_P, s \quad X) \in \mathcal{A}_P \wedge (\tau, \sqsubseteq_Q, s \quad Y) \in \mathcal{A}_Q \wedge \Sigma s \subseteq X \cup Y\}$$

$$\mathcal{P}_{PBT}\ P\ {}^X\!\|^Y\ Q\ \rho \mathrel{\widehat{=}}$$
$$\text{fillout}\,\{(\tau, \sqsubseteq, s) \mapsto \sum \Big\{\!\!\Big| f_P(\tau, \sqsubseteq_P, s \quad X).f_Q(\tau, \sqsubseteq_Q, s \quad Y) \mid \sqsubseteq\ = \ \sqsubseteq_P {}^X\!\|^Y \sqsubseteq_Q \Big|\!\!\Big\} \Big|$$
$$\Sigma s \subseteq X \cup Y\}$$

$$\mathcal{A}_{PBT}\ P \longleftarrow Q\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq_P \longleftarrow \sqsubseteq_Q, s) \mid (\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \in \mathcal{A}_P$$
$$\wedge (\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \in \mathcal{A}_Q\}$$

$$\mathcal{P}_{PBT}\ P \longleftarrow Q\ \rho(\tau, \sqsubseteq, s) \mathrel{\widehat{=}}$$
$$\sum \Big\{\!\!\Big| f_P(\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} s).f_Q(\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \mid \sqsubseteq_P \longleftarrow \sqsubseteq_Q\ = \ \sqsubseteq \Big|\!\!\Big\}$$

$$\mathcal{A}_{PBT}\ P \setminus X\ \rho \mathrel{\widehat{=}}$$
$$\{(\tau, \sqsubseteq, s) \mid \forall t \quad s \uparrow t \in \text{items} \sqsubseteq \wedge \exists \sqsubseteq' \quad \sqsubseteq' \setminus X = \sqsubseteq \wedge (\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{\setminus X} s) \in \mathcal{A}_P\}$$

$$\mathcal{P}_{PBT}\ P \setminus X\ \rho \mathrel{\widehat{=}}$$
$$\text{fillout}\{(\tau, \sqsubseteq, s) \mapsto \sum \Big\{\!\!\Big| f_P(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{\setminus X} s) \mid \sqsubseteq' \setminus X = \sqsubseteq \Big|\!\!\Big\} \mid \forall t \quad s \uparrow t \in \text{items} \sqsubseteq\}$$

$$\mathcal{A}_{PBT}\ gP\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \mid \forall t \quad s \uparrow t \in \text{items} \sqsubseteq \wedge \exists \sqsubseteq' \quad g \odot \sqsubseteq' = \sqsubseteq \wedge (\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{g} s) \in \mathcal{A}_P\}$$

$$\mathcal{P}_{PBT}\ gP\ \rho \mathrel{\widehat{=}} \text{fillout}\{(\tau, \sqsubseteq, s) \mapsto \sum \Big\{\!\!\Big| f_P(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{g} s) \mid g \odot \sqsubseteq' = \sqsubseteq \Big|\!\!\Big\} \mid \forall t \quad s \uparrow t \in \text{items} \sqsubseteq\}$$

$$\mathcal{A}_{PBT}\ P\ {}_{t}\ Q\ \rho \mathrel{\widehat{=}} \{(\tau, \sqsubseteq, s) \mid \tau \quad t \wedge (\tau, \sqsubseteq, s) \in \mathcal{A}_P\}$$
$$\cup \{(\tau, \sqsubseteq \quad (t, \tau] \otimes \langle \{\!|\,|\!\} \rangle, s \quad \prec\succ) \mid t < \tau < t + \delta \wedge (t, \sqsubseteq, s) \in \mathcal{A}_P\}$$
$$\cup \{(\tau, \sqsubseteq_P \quad (t, t + \delta) \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad \prec\succ \quad s_Q + t + \delta) \mid$$
$$\tau \quad t + \delta \wedge (t, \sqsubseteq_P, s_P) \in \mathcal{A}_P \wedge (\tau - t - \delta, \sqsubseteq_Q, s_Q) \in \mathcal{A}_Q\}$$

$$\mathcal{P}_{PBT}\; P \;\underset{t}{\;}\; Q\; \rho(\tau, \sqsubseteq, s) \mathrel{\hat=}$$

$$\begin{cases} f_P(\tau, \sqsubseteq, s) & \text{if } \tau \quad t \\ f_P((\tau, \sqsubseteq, s) \quad t) & \text{if } t < \tau < t + \delta \wedge s \quad t = \prec\succ \\ & \quad \wedge \sqsubseteq \quad t = (t, \tau] \otimes \langle\{\!|\,|\!\}\rangle \\ f_P((\tau, \sqsubseteq, s) \quad t).f_Q((\tau, \sqsubseteq, s) - t - \delta) & \text{if } \tau \quad t \wedge s \uparrow (t, t+\delta) = \prec\succ \\ & \quad \wedge \sqsubseteq \uparrow (t, t+\delta) = (t, t+\delta) \otimes \langle\{\!|\,|\!\}\rangle \\ 0 & \text{otherwise} \end{cases}$$

$$\mathcal{A}_{PBT}\; P \underset{e}{\nabla} Q\; \rho \mathrel{\hat=}$$

$$\begin{aligned} &\{(\tau, \sqsubseteq \oplus e, s) \mid (\tau, \sqsubseteq, s) \in A_P \wedge e \notin \Sigma s\} \\ &\cup\{(\tau, \sqsubseteq \oplus e \quad (t, \tau] \otimes \langle\{\!|\,|\!\}\rangle, s \quad \prec(t, e)\succ) \mid t \quad \tau < t + \delta \wedge e \notin \Sigma s \wedge (t, \sqsubseteq, s) \in A_P\} \\ &\cup\{(\tau, \sqsubseteq_P \oplus e \quad (t, t+\delta) \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad \prec(t, e)\succ \quad s_Q + t + \delta) \mid \\ &\qquad \tau \quad t + \delta \wedge e \notin \Sigma s_P \wedge (t, \sqsubseteq_P, s_P) \in A_P \wedge (\tau - t - \delta, \sqsubseteq_Q, s_Q) \in A_Q\} \end{aligned}$$

$$\mathcal{P}_{PBT}\; P \underset{e}{\nabla} Q\; \rho(\tau, \sqsubseteq, s) \mathrel{\hat=}$$

$$\begin{cases} f_P(\tau, \sqsubseteq_P, s) & \text{if } \sqsubseteq = \sqsubseteq_P \oplus e \wedge e \notin \Sigma s \\ f_P(t, \sqsubseteq_P, s_P) & \text{if } t \quad \tau < t \wedge s = s_P \quad \prec(t, e)\succ \wedge e \notin \Sigma s_P \\ & \quad \wedge \sqsubseteq = \sqsubseteq_P \oplus e \quad (t, \tau) \otimes \langle\{\!|\,|\!\}\rangle \\ f_P(t, \sqsubseteq_P, s_P).f_Q((\tau, \sqsubseteq, s) - t - \delta) & \text{if } \tau \quad t + \delta \wedge e \notin \Sigma s_P \\ & \quad \wedge s \quad t + \delta = s_P \quad \prec(t, e)\succ \\ & \quad \wedge \sqsubseteq \quad t + \delta = \sqsubseteq_P \oplus e \quad (t, t+\delta) \otimes \langle\{\!|\,|\!\}\rangle \\ 0 & \text{otherwise} \end{cases}$$

$\mathcal{F}_{PBT}\; \mu X \quad P\; \rho \mathrel{\hat=}$ the unique fixed point of the mapping $M_\delta(X, P)\rho$

$\mathcal{F}_{PBT}\; \mu X \quad P\; \rho \mathrel{\hat=}$ the unique fixed point of the mapping $M(X, P)\rho$

$\mathcal{F}_{PBT}\; \langle X_i = P_i \mid i \in I\rangle_j\; \rho \mathrel{\hat=} S_j$ where $\underline{S}$ is a fixed point of $M(\underline{X}, \underline{P})\rho$

A.4 Derived operators

$$\begin{aligned} SKIP &\mathrel{\hat=} WAIT\ 0 \\ a \xrightarrow{t} P &\mathrel{\hat=} a \xrightarrow{0} WAIT\ t\, ; P \\ P \,\square\, Q &\mathrel{\hat=} Q \,\square\, P \\ P\ {}_p\ {}_q\ Q &\mathrel{\hat=} (P \,\square\, Q)\ {}_p\sqcap_q (P \,\square\, Q) \\ P \Vert Q &\mathrel{\hat=} P\ {}^{\Sigma}\Vert^{\Sigma Q} \\ P \Vert Q &\mathrel{\hat=} Q\ {}^{\Sigma}\Vert^{\Sigma P} \\ P\ {}^X\Vert^Y\ Q &\mathrel{\hat=} Q\ {}^Y\Vert^X\ P \\ P \longrightarrow Q &\mathrel{\hat=} Q \longleftarrow P \\ P \;\overset{t}{\;}\; Q &\mathrel{\hat=} (P \,\square\, WAIT\ t\, ; trig \longrightarrow Q) \setminus trig \end{aligned}$$

and

$$P \mathbin{\underset{C}{\overleftarrow{\Vert}}} Q \mathrel{\widehat{=}} c(l(P)\ {}^{A}\overleftarrow{\Vert}^{B}\ r(Q)) \qquad P \mathbin{\underset{C}{\overrightarrow{\Vert}}} Q \mathrel{\widehat{=}} c(l(P)\ {}^{A}\overrightarrow{\Vert}^{B}\ r(Q))$$

$$\text{where } l(a) \mathrel{\widehat{=}} \begin{cases} a & \text{if } a \in C \\ l.a & \text{otherwise} \end{cases} \qquad r(a) \mathrel{\widehat{=}} \begin{cases} a & \text{if } a \in C \\ r.a & \text{otherwise} \end{cases} \qquad \begin{array}{rl} c(a) \mathrel{\widehat{=}} a & \text{if } a \in C \\ c(l.a) \mathrel{\widehat{=}} a & \text{if } a \notin C \\ c(r.a) \mathrel{\widehat{=}} a & \text{if } a \notin C \end{array}$$

$$\text{and } A \mathrel{\widehat{=}} l(\Sigma - C) \cup C \qquad B \mathrel{\widehat{=}} r(\Sigma - C) \cup C \qquad \text{and } l(\Sigma) \cap C = r(\Sigma) \cap C = \{\}$$

Appendix B

Inference Rules

B.1 Proof rules for prioritized processes

In this appendix we give a complete set of proof rules for proving specifications on BTCSP processes.

B.1.1 Auxiliary rules

Rule B.1.1 (Null specification)

$$\frac{}{P\ \mathbf{sat}_\rho\ \mathit{true}}$$

$\triangle$

Rule B.1.2 (Conjunction)

$$\frac{P\ \mathbf{sat}_\rho\ S \qquad P\ \mathbf{sat}_\rho\ T}{P\ \mathbf{sat}_\rho\ S \wedge T}$$

$\triangle$

Rule B.1.3 (Strengthen specification)

$$\frac{P\ \mathbf{sat}_\rho\ S \qquad S(\tau, \sqsubseteq, s) \Rightarrow T(\tau, \sqsubseteq, s)}{P\ \mathbf{sat}_\rho\ T}$$

$\triangle$

B.1.2 Basic processes

Rule B.1.4 ($STOP$)

$$\frac{S(\tau, [0,\tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec\succ)}{STOP\ \mathbf{sat}_\rho\ S}$$

$\triangle$

Rule B.1.5 ($SKIP$)

$$\frac{\begin{array}{l} S(\tau, [0,\tau] \otimes \langle \{\!| \quad |\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \\ t \quad \tau \Rightarrow S(\tau, [0,t] \otimes \langle \{\!| \quad |\!\}, \{\!|\,|\!\} \rangle \quad (t,\tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec(t, \quad)\succ) \end{array}}{SKIP\ \mathbf{sat}_\rho\ S}$$

$\triangle$

Rule B.1.6 ($WAIT\ t$)

$$\frac{\begin{array}{l} \tau < t \Rightarrow S(\tau, [0,\tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec\succ) \\ \tau \quad t \Rightarrow S(\tau, [0,t) \otimes \langle \{\!|\,|\!\} \rangle \quad [t,\tau] \oslash \langle \{\!| \quad |\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \\ t \quad t' \quad \tau \Rightarrow S(\tau, [0,t) \otimes \langle \{\!|\,|\!\} \rangle \quad [t,t'] \oslash \langle \{\!| \quad |\!\}, \{\!|\,|\!\} \rangle \quad (t',\tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec(t', \quad)\succ) \end{array}}{WAIT\ t\ \mathbf{sat}_\rho\ S}$$

$\triangle$

B.1.3 Sequential composition

Rule B.1.7 ($a \xrightarrow{0} P$)

$$\frac{\begin{array}{l} S(\tau, [0,\tau] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \\ P\ \mathbf{sat}_\rho\ S_P \\ S_P(\tau - t, \{0\} \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq_P, s_P) \wedge \tau \quad t \Rightarrow S(\tau, [0,t] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle \quad \sqsubseteq_P + t, (t,a) \quad s_P + t) \end{array}}{a \xrightarrow{0} P\ \mathbf{sat}_\rho\ S}$$

$\triangle$

Rule B.1.8 ($a \xrightarrow{t} P$)

$$\frac{\begin{array}{l} S(\tau, [0,\tau] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle, \prec\succ) \\ t' \quad \tau < t' + t \Rightarrow S(\tau, [0,t'] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle \quad (t',\tau] \otimes \langle \{\!|\,|\!\} \rangle, \prec(t',a)\succ) \\ P\ \mathbf{sat}_\rho\ S_P \\ S_P(\tau - t - t', \sqsubseteq_P, s_P) \wedge \tau \quad t' + t \Rightarrow \\ \qquad S(\tau, [0,t'] \otimes \langle \{\!|a|\!\}, \{\!|\,|\!\} \rangle \quad (t', t'+t) \otimes \langle \{\!|\,|\!\} \rangle \quad \sqsubseteq_P + t' + t, (t',a) \quad s_P + t' + t) \end{array}}{a \xrightarrow{t} P\ \mathbf{sat}_\rho\ S}$$

$\triangle$

Rule B.1.9 $(P \quad Q)$

$$
\frac{
\begin{array}{l}
P \ \mathbf{sat}_\rho \ S_P \\
Q \ \mathbf{sat}_\rho \ S_Q \\
\sqsubseteq' \setminus \quad = \sqsubseteq \wedge \forall t \quad s \uparrow t \in \text{items} \sqsubseteq \wedge \quad \notin \Sigma(\Uparrow_{\sqsubseteq'}^{-\setminus\checkmark} s) \wedge S_P(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus\checkmark} s) \Rightarrow S(\tau, \sqsubseteq, s) \\
\begin{pmatrix} t \quad \tau < t + \delta \wedge \sqsubseteq' \setminus \quad = \sqsubseteq \wedge \forall t' \quad s \uparrow t' \in \text{items} \sqsubseteq \\ \wedge\, begin((\Uparrow_{\sqsubseteq'}^{-\setminus\checkmark} s) \quad) = t \wedge S_P(t, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus\checkmark} s) \end{pmatrix} \Rightarrow S(\tau, \sqsubseteq \quad (t, \tau] \otimes \langle\{\!|\,|\!\}\rangle, s) \\
\begin{pmatrix} t \quad \tau - \delta \wedge \sqsubseteq'_P \setminus \quad = \sqsubseteq_P \wedge \forall t' \quad s_P \uparrow t' \in \text{items} \sqsubseteq_P \\ \wedge\, begin((\Uparrow_{\sqsubseteq'_P}^{-\setminus\checkmark} s_P) \quad) = t \wedge S_P(t, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{-\setminus\checkmark} s) \wedge S_Q(\tau - (t + \delta), \sqsubseteq_Q, s_Q) \end{pmatrix} \Rightarrow \\
\qquad S(\tau, \sqsubseteq_P \quad (t, t + \delta) \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad s_Q + t + \delta)
\end{array}
}{P \quad Q \ \mathbf{sat}_\rho \ S}
$$

$\triangle$

Rule B.1.10 $(WAIT\ t\,; P)$

$$
\frac{
\begin{array}{l}
\tau < t \Rightarrow S(\tau, [0, \tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec\succ) \\
P \ \mathbf{sat}_\rho \ S_P \\
S_P(\tau, \sqsubseteq, s) \Rightarrow S(\tau + t, \sqsubseteq + t, s + t)
\end{array}
}{WAIT\ t\,; P \ \mathbf{sat}_\rho \ S}
$$

$\triangle$

B.1.4 Nondeterministic choice

Rule B.1.11 $(P \sqcap Q)$

$$
\frac{
\begin{array}{l}
P \ \mathbf{sat}_\rho \ S_P \\
Q \ \mathbf{sat}_\rho \ S_Q \\
S_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s)
\end{array}
}{P \sqcap Q \ \mathbf{sat}_\rho \ S}
$$

$\triangle$

Rule B.1.12 $(\ {}_{i \in I} P_i)$

$$
\frac{
\begin{array}{l}
\forall i \in I \quad P_i \ \mathbf{sat}_\rho \ S_i \\
\forall i \in I \quad S_i(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s)
\end{array}
}{{}_{i \in I} P_i \ \mathbf{sat}_\rho \ S}
$$

$\triangle$

B.1.5 External choice

Rule B.1.13 $(P \mathbin{|\!\Box} Q)$

$$
\frac{
\begin{array}{l}
P \ \mathbf{sat}_\rho \ S_P \\
Q \ \mathbf{sat}_\rho \ S_Q \\
S_P(\tau, \sqsubseteq_P, \prec\succ) \wedge S_Q(\tau, \sqsubseteq_Q, \prec\succ) \Rightarrow S(\tau, \sqsubseteq_P \mathbin{|\!\Box} \sqsubseteq_Q, \prec\succ) \\
\begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(t, \sqsubseteq_Q, \prec\succ) \\ \wedge\ (s \uparrow t \sqsupset_P (t, \{\!|\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_Q) \end{pmatrix} \Rightarrow S(\tau, \sqsubseteq_P \mathbin{|\!\Box} \sqsubseteq_Q, s) \\
\begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge S_P(t, \sqsubseteq_P, \prec\succ) \\ \wedge\ S_Q(\tau, \sqsubseteq_Q, s) \wedge s \uparrow t \not\sqsupset_P (t, \{\!|\!\}) \end{pmatrix} \Rightarrow S(\tau, \sqsubseteq_P \mathbin{|\!\Box} \sqsubseteq_Q, s)
\end{array}
}{
P \mathbin{|\!\Box} Q \ \mathbf{sat}_\rho \ S
}
$$

$\triangle$

Rule B.1.14 $(P \mathbin{\Box\!|} Q)$

$$
\frac{
\begin{array}{l}
P \ \mathbf{sat}_\rho \ S_P \\
Q \ \mathbf{sat}_\rho \ S_Q \\
S_P(\tau, \sqsubseteq_P, \prec\succ) \wedge S_Q(\tau, \sqsubseteq_Q, \prec\succ) \Rightarrow S(\tau, \sqsubseteq_P \mathbin{\Box\!|} \sqsubseteq_Q, \prec\succ) \\
\begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge S_P(\tau, \sqsubseteq_P, s) \\ \wedge\ S_Q(t, \sqsubseteq_Q, \prec\succ) \wedge s \uparrow t \not\sqsupset_Q (t, \{\!|\!\}) \end{pmatrix} \Rightarrow S(\tau, \sqsubseteq_P \mathbin{\Box\!|} \sqsubseteq_Q, s) \\
\begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\, s = t \wedge S_P(t, \sqsubseteq_P, \prec\succ) \wedge S_Q(\tau, \sqsubseteq_Q, s) \\ \wedge\ (s \uparrow t \sqsupset_Q (t, \{\!|\!\}) \vee s \uparrow t \notin \text{items} \sqsubseteq_P) \end{pmatrix} \Rightarrow S(\tau, \sqsubseteq_P \mathbin{\Box\!|} \sqsubseteq_Q, s)
\end{array}
}{
\mathrm{P} \mathbin{\Box\!|} Q \ \mathbf{sat}_\rho \ S
}
$$

$\triangle$

Rule B.1.15 $(c?a : A \xrightarrow{t_a} P_a)$

$$
\frac{
\begin{array}{l}
\forall\, a \in A \quad P_a \ \mathbf{sat}_\rho \ S_a \\
S(\tau, [0, \tau] \otimes (\{\!|c?a|\!\}, \{\!|\!\}), \prec\succ) \\
\forall\, a \in A \quad t \quad \tau < t + t_a \Rightarrow S(\tau, [0, t] \otimes \langle \{\!|c?a|\!\}, \{\!|\!\}\rangle \quad (t, \tau] \otimes \langle \{\!|\!\}\rangle, \prec(t, a)\succ) \\
\forall\, a \in A \quad \tau \quad t + t_a \wedge S_a(\tau - t - t_a, \sqsubseteq, s) \Rightarrow \\
\qquad\qquad S(\tau, [0, t] \otimes \langle \{\!|c?a|\!\}, \{\!|\!\}\rangle \quad (t, t + t_a) \odot \langle \{\!|\!\}\rangle \quad \sqsubseteq + t + t_a, \prec(t, a)\succ \quad s + t + t_a)
\end{array}
}{
c?a : A \xrightarrow{t_a} P_a \ \mathbf{sat}_\rho \ S
}
$$

$\triangle$

B.1.6 Parallel composition

Rule B.1.16 $(P \mathbin{\#} Q)$

$$
\frac{
\begin{array}{l}
P \ \mathbf{sat}_\rho \ S_P \\
Q \ \mathbf{sat}_\rho \ S_Q \\
S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \mathbin{\#} \sqsubseteq_Q, s)
\end{array}
}{
P \mathbin{\#} Q \ \mathbf{sat}_\rho \ S
}
$$

$\triangle$

Rule B.1.17 $(P \mathbin{\Vert\!\!\!\!\rightarrow} Q)$

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho\; S_P \\ Q \;\mathbf{sat}_\rho\; S_Q \\ S_P(\tau, \sqsubseteq_P, s) \wedge S_Q(\tau, \sqsubseteq_Q, s) \Rightarrow S(\tau, \sqsubseteq_P \mathbin{\Vert\!\!\!\!\rightarrow} \sqsubseteq_Q, s) \end{array}}{P \mathbin{\Vert\!\!\!\!\rightarrow} Q \;\mathbf{sat}_\rho\; S}$$

$\triangle$

Rule B.1.18 $(P \;{}^{X}\!\!\mathbin{\leftarrow\!\!\!\!\Vert}^{Y}\; Q)$

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho\; S_P \\ Q \;\mathbf{sat}_\rho\; S_Q \\ S_P(\tau, \sqsubseteq_P, s \quad X) \wedge S_Q(\tau, \sqsubseteq_Q, s \quad Y) \wedge \Sigma s \subseteq X \cup Y \Rightarrow S(\tau, \sqsubseteq_P {}^{X}\!\!\mathbin{\leftarrow\!\!\!\!\Vert}^{Y} \sqsubseteq_Q, s) \end{array}}{P \;{}^{X}\!\!\mathbin{\leftarrow\!\!\!\!\Vert}^{Y}\; Q \;\mathbf{sat}_\rho\; S}$$

$\triangle$

Rule B.1.19 $(P \;{}^{X}\!\mathbin{\Vert\!\!\!\!\rightarrow}^{Y}\; Q)$

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho\; S_P \\ Q \;\mathbf{sat}_\rho\; S_Q \\ S_P(\tau, \sqsubseteq_P, s \quad X) \wedge S_Q(\tau, \sqsubseteq_Q, s \quad Y) \wedge \Sigma s \subseteq X \cup Y \Rightarrow S(\tau, \sqsubseteq_P {}^{X}\!\mathbin{\Vert\!\!\!\!\rightarrow}^{Y} \sqsubseteq_Q, s) \end{array}}{P \;{}^{X}\!\mathbin{\Vert\!\!\!\!\rightarrow}^{Y}\; Q \;\mathbf{sat}_\rho\; S}$$

$\triangle$

B.1.7 Interleaving

Rule B.1.20 $(P \longleftarrow Q)$

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho\; S_P \\ Q \;\mathbf{sat}_\rho\; S_Q \\ S_P(\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \wedge S_Q(\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \Rightarrow S(\tau, \sqsubseteq_P \longleftarrow \sqsubseteq_Q, s) \end{array}}{P \longleftarrow Q \;\mathbf{sat}_\rho\; S}$$

$\triangle$

Rule B.1.21 $(P \longrightarrow Q)$

$$\frac{\begin{array}{l} P \;\mathbf{sat}_\rho\; S_P \\ Q \;\mathbf{sat}_\rho\; S_Q \\ S_P(\tau, \sqsubseteq_P, \Downarrow_{\sqsubseteq_Q, \sqsubseteq_P} s) \wedge S_Q(\tau, \sqsubseteq_Q, \Uparrow_{\sqsubseteq_Q, \sqsubseteq_P} s) \Rightarrow S(\tau, \sqsubseteq_P \longrightarrow \sqsubseteq_Q, s) \end{array}}{P \longrightarrow Q \;\mathbf{sat}_\rho\; S}$$

$\triangle$

Rule B.1.22 ($P \overleftarrow{\|}_C Q$)

$$\frac{\begin{array}{l} P \ \mathbf{sat}_\rho \ S_P \\ Q \ \mathbf{sat}_\rho \ S_Q \\ S_P(\tau, \sqsubseteq_P, \Uparrow_C^{\sqsubseteq_P, \sqsubseteq_Q} s) \wedge S_Q(\tau, \sqsubseteq_Q, \Downarrow_C^{\sqsubseteq_P, \sqsubseteq_Q} s) \Rightarrow S(\tau, \sqsubseteq_P \overleftarrow{\|}_C \sqsubseteq_Q, s) \end{array}}{P \overleftarrow{\|}_C Q \ \mathbf{sat}_\rho \ S}$$

△

Rule B.1.23 ($P \overrightarrow{\|}_C Q$)

$$\frac{\begin{array}{l} P \ \mathbf{sat}_\rho \ S_P \\ Q \ \mathbf{sat}_\rho \ S_Q \\ S_P(\tau, \sqsubseteq_P, \Downarrow_C^{\sqsubseteq_Q, \sqsubseteq_P} s) \wedge S_Q(\tau, \sqsubseteq_Q, \Uparrow_C^{\sqsubseteq_Q, \sqsubseteq_P} s) \Rightarrow S(\tau, \sqsubseteq_P \overrightarrow{\|}_C \sqsubseteq_Q, s) \end{array}}{P \overrightarrow{\|}_C Q \ \mathbf{sat}_\rho \ S}$$

△

B.1.8 Abstraction and renaming

Rule B.1.24 ($P \setminus X$)

$$\frac{\begin{array}{l} P \ \mathbf{sat}_\rho \ S_P \\ \sqsubseteq' \setminus X = \sqsubseteq \wedge \forall t \quad s \uparrow t \in \text{items} \sqsubseteq \wedge S_P(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq'}^{\setminus X} s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{P \setminus X \ \mathbf{sat}_\rho \ S}$$

△

Rule B.1.25 ($P \setminus X$)

$$\frac{P \ \mathbf{sat}_\rho \ \mathsf{internal}\ A \Rightarrow S(\tau, \sqsubseteq, s)}{P \setminus A \ \mathbf{sat}_\rho \ S} \Big[S \text{ is } A\text{-independent} \Big]$$

△

Rule B.1.26 ($f(P)$)

$$\frac{\begin{array}{l} P \ \mathbf{sat}_\rho \ S_P \\ f \sqsubseteq' = \sqsubseteq \wedge \forall t \quad s \uparrow t \in \text{items} \sqsubseteq \wedge S_P(\tau, \sqsubseteq', \Uparrow_{\sqsubseteq}^{f} s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{f(P) \ \mathbf{sat}_\rho \ S}$$

△

B.1.9 Transfer operators

Rule B.1.27 $(P \;{}^{t}\; Q)$

$$
\begin{array}{l}
P\ \mathbf{sat}_\rho\ S_P \\
Q\ \mathbf{sat}_\rho\ S_Q \\
\tau \quad t \wedge S_P(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \\
\mathit{begin}\ s \quad t \wedge S_P(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \\
t < \tau < t + \delta \wedge S_P(t, \sqsubseteq, \prec\succ) \Rightarrow S(\tau, \sqsubseteq \quad (t, \tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec\succ) \\
\tau \quad t + \delta \wedge S_P(t, \sqsubseteq_P, \prec\succ) \wedge S_Q(\tau - t - \delta, \sqsubseteq_Q, s_Q) \Rightarrow \\
\quad S(\tau, \sqsubseteq_P \quad (t, t+\delta) \otimes \langle\{\!|\,|\!\}\rangle \quad s + t + \delta, \prec\succ \quad s_Q + t + \delta) \\
\hline
P \;{}^{t}\; Q\ \mathbf{sat}_\rho\ S
\end{array}
$$

$\triangle$

Rule B.1.28 $(P \;{}_{t}\; Q)$

$$
\begin{array}{l}
P\ \mathbf{sat}_\rho\ S_P \\
Q\ \mathbf{sat}_\rho\ S_Q \\
\tau \quad t \wedge S_P(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \\
t < \tau < t + \delta \wedge S_P(t, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq \quad (t, \tau] \otimes \langle\{\!|\,|\!\}\rangle, s \quad \prec\succ) \\
\tau \quad t + \delta \wedge S_P(t, \sqsubseteq_P, s_P) \wedge S_Q(\tau - t - \delta, \sqsubseteq_Q, s_Q) \Rightarrow \\
\quad S(\tau, \sqsubseteq_P \quad (t, t+\delta) \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad \prec\succ \quad s_Q + t + \delta) \\
\hline
P \;{}_{t}\; Q\ \mathbf{sat}_\rho\ S
\end{array}
$$

$\triangle$

Rule B.1.29 $(P \underset{e}{\nabla} Q)$

$$
\begin{array}{l}
P\ \mathbf{sat}_\rho\ S_P \\
Q\ \mathbf{sat}_\rho\ S_Q \\
e \notin \Sigma s \wedge S_P(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq \oplus e, s) \\
t \quad \tau < t + \delta \wedge e \notin \Sigma s \wedge S_P(t, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq \oplus e \quad (t, \tau] \otimes \langle\{\!|\,|\!\}\rangle, s \quad \prec(t, e)\succ) \\
\tau \quad t + \delta \wedge e \notin \Sigma s_P \wedge S_P(t, \sqsubseteq_P, s_P) \wedge S_Q(\tau - t - \delta, \sqsubseteq_Q, s_Q) \Rightarrow \\
\quad S(\tau, \sqsubseteq_P \quad (t, t+\delta) \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq_Q + t + \delta, s_P \quad \prec(t, e)\succ \quad s_Q + t + \delta) \\
\hline
P \underset{e}{\nabla} Q\ \mathbf{sat}_\rho\ S
\end{array}
$$

$\triangle$

B.1.10 Recursion

Rule B.1.30 $(\mu X \quad P)$

$$
\begin{array}{l}
X\ \mathbf{sat}_\rho\ S \Rightarrow P\ \mathbf{sat}_\rho\ S \\
\hline
\mu X \quad P\ \mathbf{sat}_\rho\ S
\end{array}
$$

$\triangle$

Rule B.1.31 ($\mu X \quad P$)

$$\frac{X\ \mathbf{sat}_\rho\ (S((\tau, \sqsubseteq, s) - \delta) \wedge \mathit{begin}\ s \quad \delta \wedge \sqsubseteq \quad \delta = [0,\delta) \otimes \langle\{\![\,]\!\}\rangle) \Rightarrow P\ \mathbf{sat}_\rho\ S(\tau, \sqsubseteq, s)}{\mu X \quad P\ \mathbf{sat}_\rho\ S}$$

$\triangle$

Rule B.1.32 ($\langle X_i = p_i \mid i \in I\rangle_j$)

$$\frac{(\forall i : I \quad X_i\ \mathbf{sat}_\rho\ S_i) \Rightarrow \forall j : I \quad P_i\ \mathbf{sat}_\rho\ S_i}{\langle X_i = P_i\rangle_j\ \mathbf{sat}_\rho\ S_j}$$

$\triangle$

B.2 Proof rules for unprobabilistic specifications on probabilistic processes

The following rule can be used to reduce a proof obligation in $\mathcal{M}_{PTB}$ to a proof obligation in $\mathcal{M}_{TB}$:

Rule B.2.1 (Abstraction)

$$\frac{\varphi_P^{(B)}(P)\ \mathbf{sat}_{\rho'}\ S \text{ in } \mathcal{M}_{TB}}{P\ \mathbf{sat}_\rho\ S \text{ in } \mathcal{M}_{PTB}}\left[\forall X : \mathit{VAR} \quad \pi_1(\rho\ X\ ) = \rho'\ X\ \right]$$

$\triangle$

All the rules from appendix B.1 can be used for proving that probabilistic processes satisfy hard specifications, except the rules for nondeterministic choice should be replaced by the following rules for probabilistic choice:

Rule B.2.2 ($P\ {}_p\sqcap_q\ Q$)

$$\frac{\begin{array}{l} P\ \mathbf{sat}_\rho\ S_P \\ Q\ \mathbf{sat}_\rho\ S_Q \\ S_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{P\ {}_p\sqcap_q\ Q\ \mathbf{sat}_\rho\ S}$$

$\triangle$

Rule B.2.3 ($_{i \in I}[p_i]P_i$)

$$\frac{\begin{array}{l} \forall i \in I \quad P_i\ \mathbf{sat}_\rho\ S_i \\ \forall i \in I \quad S_i(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \end{array}}{_{i \in I}[p_i]P_i\ \mathbf{sat}_\rho\ S}$$

$\triangle$

Rule B.2.4 $(P \;{}_p\;\; {}_q\; Q)$

$$\begin{array}{l} P \,\square\, Q \;\mathbf{sat}_\rho\; S \\ P \,\square\!\!\!\mid\, Q \;\mathbf{sat}_\rho\; S \\ \hline P \;{}_p\;\; {}_q\; Q \;\mathbf{sat}_\rho\; S \end{array}$$

$\triangle$

B.3 Proof rules for probabilistic specifications

B.3.1 Basic processes

Rule B.3.1 $(STOP)$

$$\begin{array}{l} S(\tau, [0,\tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec\succ) \\ \hline STOP \;\mathbf{sat}_\rho^{\geqslant 1}\; S \end{array}$$

$\triangle$

Rule B.3.2 $(WAIT\ t)$

$$\begin{array}{l} \tau < t \Rightarrow S(\tau, [0,\tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec\succ) \\ \tau \quad t \Rightarrow S(\tau, [0,t) \otimes \langle\{\!|\,|\!\}\rangle \quad [t,\tau] \otimes \langle\{\!|\,|\!\}, \{\!|\quad|\!\}\rangle, \prec\succ) \\ t \quad t' \quad \tau \Rightarrow S(\tau, [0,t) \otimes \langle\{\!|\,|\!\}\rangle \quad [t,t'] \otimes \langle\{\!|\,|\!\}, \{\!|\quad|\!\}\rangle \quad (t',\tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec(t', \quad)\succ) \\ \hline WAIT\ t \;\mathbf{sat}_\rho^{\geqslant 1}\; S \end{array}$$

$\triangle$

Rule B.3.3 $(SKIP)$

$$\begin{array}{l} S(\tau, [0,\tau] \otimes \langle\{\!|\,|\!\}, \{\!|\quad|\!\}\rangle, \prec\succ) \\ t \quad \tau \Rightarrow S(\tau, [0,t] \otimes \langle\{\!|\,|\!\}, \{\!|\quad|\!\}\rangle \quad (t,\tau] \otimes \langle\{\!|\,|\!\}\rangle, \prec(t, \quad)\succ) \\ \hline SKIP \;\mathbf{sat}_\rho^{\geqslant 1}\; S \end{array}$$

$\triangle$

B.3.2 Sequential composition

Rule B.3.4 $(a \xrightarrow{0} P)$

$$\begin{array}{l} P \;\mathbf{sat}_\rho^{\geqslant p}\; S_P \mid G_P \\ G(\tau,\ [0,\tau] \otimes \langle\{\!|a|\!\}, \{\!|\,|\!\}\rangle,\ \prec\succ) \Rightarrow S(\tau,\ [0,\tau] \otimes \langle\{\!|a|\!\}, \{\!|\,|\!\}\rangle,\ \prec\succ) \\ S_P(\tau,\ 0 \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq,\ s) \wedge G_P(\tau,\ 0 \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq,\ s) \Rightarrow \\ \qquad \left(\begin{array}{l} S(\tau + t,\ [0,t] \otimes \langle\{\!|a|\!\}, \{\!|\,|\!\}\rangle \quad \sqsubseteq + t,\ (t,a) \quad s + t) \\ \wedge\ G(\tau + t,\ [0,t] \otimes \langle\{\!|a|\!\}, \{\!|\,|\!\}\rangle \quad \sqsubseteq + t,\ (t,a) \quad s + t) \end{array}\right) \\ G(\tau + t,\ [0,t] \otimes \langle\{\!|a|\!\}, \{\!|\,|\!\}\rangle \quad \sqsubseteq + t,\ (t,a) \quad s + t) \Rightarrow G_P(\tau,\ 0 \otimes \langle\{\!|\,|\!\}\rangle \quad \sqsubseteq,\ s) \\ \hline a \xrightarrow{0} P \;\mathbf{sat}_\rho^{\geqslant p}\; S \mid G \end{array}$$

$\triangle$

Rule B.3.5 $(a \xrightarrow{t} P)$

$$
\begin{array}{l}
P\ \mathbf{sat}_\rho^{\geqslant p}\ S_P \mid G_P \\
G(\tau,\ [0,\tau] \otimes \langle \{\!\!\{a\}\!\!\}, \{\!\!\{\}\!\!\} \rangle,\ \prec\succ) \Rightarrow S(\tau,\ [0,\tau] \otimes \langle \{\!\!\{a\}\!\!\}, \{\!\!\{\}\!\!\} \rangle,\ \prec\succ) \\
t' \quad \tau < t + t' \wedge G(\tau,\ [0,t'] \otimes \langle \{\!\!\{a\}\!\!\}, \{\!\!\{\}\!\!\} \rangle \quad (t',\tau] \otimes \langle \{\!\!\{\}\!\!\} \rangle,\ \prec(t',a)\succ) \Rightarrow \\
\quad S(\tau,\ [0,t'] \otimes \langle \{\!\!\{a\}\!\!\}, \{\!\!\{\}\!\!\} \rangle \quad (t',\tau] \otimes \langle \{\!\!\{\}\!\!\} \rangle,\ \prec(t',a)\succ) \\
S_P(\tau,\ 0 \otimes \langle \{\!\!\{\}\!\!\} \rangle \quad \sqsubseteq,\ s) \wedge G_P(\tau,\ 0 \otimes \langle \{\!\!\{\}\!\!\} \rangle \quad \sqsubseteq,\ s) \Rightarrow \\
\quad \begin{pmatrix} S(\tau + t + t',\ [0,t'] \otimes \langle \{\!\!\{a\}\!\!\}, \{\!\!\{\}\!\!\} \rangle \quad (t', t+t') \otimes \langle \{\!\!\{\}\!\!\} \rangle \quad \sqsubseteq + t + t',\ (t',a) \quad s + t + t') \\ \wedge\ G(\tau + t + t',\ [0,t'] \otimes \langle \{\!\!\{a\}\!\!\}, \{\!\!\{\}\!\!\} \rangle \quad (t', t+t') \otimes \langle \{\!\!\{\}\!\!\} \rangle \quad \sqsubseteq + t + t',\ (t',a) \quad s + t + t') \end{pmatrix} \\
G(\tau + t + t',\ [0,t'] \otimes \langle \{\!\!\{a\}\!\!\}, \{\!\!\{\}\!\!\} \rangle \quad (t', t+t') \otimes \langle \{\!\!\{\}\!\!\} \rangle \quad \sqsubseteq + t + t',\ (t',a) \quad s + t + t') \Rightarrow \\
\quad G_P(\tau,\ 0 \otimes \langle \{\!\!\{\}\!\!\} \rangle \quad \sqsubseteq,\ s) \\
\hline
a \xrightarrow{t} P\ \mathbf{sat}_\rho^{\geqslant p}\ S \mid G
\end{array}
$$

△

Rule B.3.6 $(P \quad Q)$

$$
\begin{array}{ll}
\forall i & P\ \mathbf{sat}_\rho^{\geqslant p_i}\ S_{P,i} \mid G_{P,i} \wedge \quad \notin \Sigma s \wedge \text{internal} \\
\forall i & P\ \mathbf{sat}_\rho^{\geqslant p_i'}\ S'_{P,i} \mid G'_{P,i} \wedge \text{internal} \quad \wedge \text{time of first} \quad = \tau \\
\forall i & Q\ \mathbf{sat}_\rho^{\geqslant q_i}\ S_{Q,i} \mid G_{Q,i} \\
\forall i & S_{P,i} \wedge G_{P,i} \wedge \text{internal} \quad \wedge \quad \notin \Sigma s \Rightarrow \\
 & \quad S(\tau, \sqsubseteq \setminus \quad, s) \wedge G(\tau, \sqsubseteq \setminus \quad, s) \\
\forall i & \begin{pmatrix} S'_{P,i}(t, \sqsubseteq, s) \wedge G'_{P,i}(t, \sqsubseteq, s) \wedge (\text{internal} \quad)(t, \sqsubseteq, s) \\ \wedge\ \mathit{begin}(s \quad) = t \wedge t \quad \tau < t + \delta \end{pmatrix} \Rightarrow \\
 & \quad \begin{pmatrix} S((t, \sqsubseteq \setminus \quad, s \setminus \quad) \quad \mathsf{empty}_{(t,\tau]}) \\ \wedge\ G((t, \sqsubseteq \setminus \quad, s \setminus \quad) \quad \mathsf{empty}_{(t,\tau]}) \end{pmatrix} \\
\forall i & \begin{pmatrix} S'_{P,i}(t, \sqsubseteq, s) \wedge G'_{P,i}(t, \sqsubseteq, s) \wedge S_{Q,i}(b_Q) \wedge G_{Q,i}(b_Q) \\ \wedge\ (\text{internal} \quad)(t, \sqsubseteq, s) \wedge \mathit{begin}(s \quad) = t \end{pmatrix} \Rightarrow \\
 & \quad \begin{pmatrix} S((t, \sqsubseteq \setminus \quad, s \setminus \quad) \quad \mathsf{empty}_{(t,t+\delta)} \quad b_Q + t + \delta) \\ \wedge\ G((t, \sqsubseteq \setminus \quad, s \setminus \quad) \quad \mathsf{empty}_{(t,t+\delta)} \quad b_Q + t + \delta) \end{pmatrix} \\
\forall i & G(\tau, \sqsubseteq \setminus \quad, s) \wedge (\text{internal} \quad)(\tau, \sqsubseteq, s) \wedge \quad \notin \Sigma s \Rightarrow G_{P,i}(\tau, \sqsubseteq, s) \\
\forall i & \begin{pmatrix} G((t, \sqsubseteq \setminus \quad, s \setminus \quad) \quad \mathsf{empty}_{(t,\tau]}) \\ \wedge\ (\text{internal} \quad)(t, \sqsubseteq, s) \\ \wedge\ \mathit{begin}(s \quad) = t \wedge t \quad \tau < t + \delta \end{pmatrix} \Rightarrow G'_{P,i}(t, \sqsubseteq, s) \\
\forall i & \begin{pmatrix} G((t, \sqsubseteq \setminus \quad, s \setminus \quad) \quad \mathsf{empty}_{(t,t+\delta)} \quad b_Q + t + \delta) \\ \wedge\ (\text{internal} \quad)(t, \sqsubseteq, s) \wedge \mathit{begin}(s \quad) = t \end{pmatrix} \Rightarrow \\
 & \quad G'_{P,i}(t, \sqsubseteq, s) \wedge G_{Q,i}(b_Q) \\
\hline
\multicolumn{2}{l}{P \quad Q\ \mathbf{sat}_\rho^{\geqslant \Sigma_i p_i}\ S \mid G}
\end{array}
\quad
\begin{bmatrix} \langle \hat{S}_i(\tau, \sqsubseteq, s) \rangle \\ \quad \mathit{disjoint} \\ \langle \hat{S}'_i(\tau, \sqsubseteq, s) \rangle \\ \quad \mathit{disjoint} \\ \Sigma_i\, p'_i q_i \quad \Sigma_i\, p_i \end{bmatrix}
$$

where i ranges over some set I and

$$
\begin{array}{l}
\hat{S}_i(\tau, \sqsubseteq, s) \ \widehat{=}\ (\text{internal} \quad)(\tau, \sqsubseteq, s) \wedge \quad \notin \Sigma s \wedge S_{P,i}(\tau, \sqsubseteq, s) \wedge G_{P,i}(\tau, \sqsubseteq, s) \\
\hat{S}'_i(\tau, \sqsubseteq, s) \ \widehat{=}\ (\text{internal} \quad)(\tau, \sqsubseteq, s) \wedge \mathit{begin}(s \quad) = t \wedge S'_{P,i}(\tau, \sqsubseteq, s) \wedge G'_{P,i}(\tau, \sqsubseteq, s)
\end{array}
$$

△

Rule B.3.7 ($WAIT\ t\ ;\ P$)

$$
\begin{array}{l}
P\ \mathsf{sat}_{\rho}^{\geqslant p}\ S_P \mid G_P \\
\tau < t \wedge G(\tau,\ [0,\tau] \otimes \langle\{\!\!\}\rangle,\ \prec\succ) \Rightarrow S(\tau,\ [0,\tau] \otimes \langle\{\!\!\}\rangle,\ \prec\succ) \\
S_P(\tau, \sqsubseteq, s) \wedge G_P(\tau, \sqsubseteq, s) \Rightarrow \\
\quad S(\tau + t,\ [0,t) \otimes \langle\{\!\!\}\rangle\ \ \sqsubseteq + t,\ s + t) \wedge G(\tau + t,\ [0,t) \otimes \langle\{\!\!\}\rangle\ \ \sqsubseteq + t,\ s + t) \\
G(\tau + t,\ [0,t) \otimes \langle\{\!\!\}\rangle\ \ \sqsubseteq + t,\ s + t) \Rightarrow G_P(\tau, \sqsubseteq, s) \\
\hline
WAIT\ t\ ;\ P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S \mid G
\end{array}
$$

△

B.3.3 Probabilistic choice

Rule B.3.8 ($P\ {}_p\sqcap_q\ Q$, **unconditional specifications**)

$$
\begin{array}{l}
P\ \mathsf{sat}_{\rho}^{\geqslant p'}\ S_P \\
Q\ \mathsf{sat}_{\rho}^{\geqslant q'}\ S_Q \\
S_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \\
\hline
P\ {}_p\sqcap_q\ Q\ \mathbf{sat}_{\rho}^{\geqslant p \cdot p' + q \cdot q'}\ S
\end{array}
$$

△

Rule B.3.9 ($P\ {}_p\sqcap_q\ Q$, **conditional specifications**)

$$
\begin{array}{l}
P\ \mathsf{sat}_{\rho}^{\geqslant p'}\ S_P \mid G_P \\
Q\ \mathsf{sat}_{\rho}^{\geqslant p'}\ S_Q \mid G_Q \\
S_P(\tau, \sqsubseteq, s) \wedge G_P(\tau, \sqsubseteq, s) \vee S_Q(\tau, \sqsubseteq, s) \wedge G_Q(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\
G(\tau, \sqsubseteq, s) \Rightarrow G_P(\tau, \sqsubseteq, s) \wedge G_Q(\tau, \sqsubseteq, s) \\
\hline
P\ {}_p\sqcap_q\ Q\ \mathsf{sat}_{\rho}^{\geqslant p'}\ S \mid G
\end{array}
$$

△

Rule B.3.10 (${}_{i \in I}[p_i]P_i$, **unconditional specifications**)

$$
\begin{array}{ll}
\forall i & P_i\ \mathbf{sat}_{\rho}^{\geqslant q_i}\ S_i \\
\forall i & S_i(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \\
\hline
\multicolumn{2}{l}{{}_{i \in I}[p_i]P_i\ \mathbf{sat}_{\rho}^{\geqslant \Sigma p_i q_i}\ S}
\end{array}
$$

△

Rule B.3.11 (${}_{i \in I}[p_i]P_i$, **conditional specifications**)

$$
\begin{array}{ll}
\forall i & P_i\ \mathsf{sat}_{\rho}^{\geqslant p}\ S_i \mid G_i \\
\forall i & S_i(\tau, \sqsubseteq, s) \wedge G_i(\tau, \sqsubseteq, s) \Rightarrow S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\
\forall i & G(\tau, \sqsubseteq, s) \Rightarrow G_i(\tau, \sqsubseteq, s) \\
\hline
\multicolumn{2}{l}{{}_{i \in I}[p_i]P_i\ \mathbf{sat}_{\rho}^{\geqslant p}\ S}
\end{array}
$$

△

B.3.4 External choice

Rule B.3.12 ($P \mathbin{\square} Q$)

$$
\dfrac{
\begin{array}{ll}
\forall i & P \ \mathbf{sat}_\rho^{\geq p_i} \ S_{P,i} \mid G_{P,i} \\
\forall i & Q \ \mathbf{sat}_\rho^{\geq q_i} \ S_{Q,i} \mid G_{Q,i} \\
\forall i & \begin{pmatrix} S_{P,i}(\tau, \sqsubseteq_P, \prec\succ) \wedge G_{P,i}(\tau, \sqsubseteq_P, \prec\succ) \\ \wedge\ S_{Q,i}(\tau, \sqsubseteq_Q, \prec\succ) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, \prec\succ) \end{pmatrix} \Rightarrow \\
& \quad S(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, \prec\succ) \wedge G(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, \prec\succ) \\
\forall i & \begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\ s = t \\ \wedge\ (s \uparrow t \sqsupset_P (t, \{\!\|\!\}) \vee s \uparrow t \notin \mathrm{items} \sqsubseteq_Q) \\ \wedge\ S_{P,i}(\tau, \sqsubseteq_P, s) \wedge G_{P,i}(\tau, \sqsubseteq_P, s) \\ \wedge\ S_{Q,i}(t, \sqsubseteq_Q, \prec\succ) \wedge G_{Q,i}(t, \sqsubseteq_Q, \prec\succ) \end{pmatrix} \Rightarrow \\
& \quad S(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, s) \wedge G(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, s) \\
\forall i & \begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\ s = t \wedge s \uparrow t \not\sqsupset_P (t, \{\!\|\!\}) \\ \wedge\ S_{P,i}(t, \sqsubseteq_P, \prec\succ) \wedge G_{P,i}(t, \sqsubseteq_P, \prec\succ) \\ \wedge\ S_{Q,i}(\tau, \sqsubseteq_Q, s) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s) \end{pmatrix} \Rightarrow \\
& \quad S(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, s) \wedge G(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, s) \\
\forall i & G(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, \prec\succ) \Rightarrow G_{P,i}(\tau, \sqsubseteq_P, \prec\succ) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, \prec\succ) \\
\forall i & \begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\ s = t \wedge G(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, s) \\ \wedge\ (s \uparrow t \sqsupset_P (t, \{\!\|\!\}) \vee s \uparrow t \notin \mathrm{items} \sqsubseteq_Q) \end{pmatrix} \Rightarrow \\
& \quad G_{P,i}(\tau, \sqsubseteq_P, s) \wedge G_{Q,i}(t, \sqsubseteq_Q, \prec\succ) \\
\forall i & \begin{pmatrix} s \neq \prec\succ \wedge \mathit{begin}\ s = t \\ \wedge\ s \uparrow t \not\sqsupset_P (t, \{\!\|\!\}) \wedge G(\tau, \sqsubseteq_P \mathbin{\square} \sqsubseteq_Q, s) \end{pmatrix} \Rightarrow \begin{pmatrix} G_{P,i}(t, \sqsubseteq_P, \prec\succ) \\ \wedge\ G_{Q,i}(\tau, \sqsubseteq_Q, s) \end{pmatrix}
\end{array}
}{
P \mathbin{\square} Q \ \mathbf{sat}_\rho^{\geq \Sigma_i p_i q_i} \ S \mid G
}
\left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ \mathit{disjoint} \end{array} \right]
$$

where i ranges over some set I and

$$
\begin{aligned}
\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\widehat{=}}\ & S_{P,i}(\tau, \sqsubseteq_P, \prec\succ) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, \prec\succ) \wedge s = \prec\succ \\
& \vee S_{P,i}(\tau, \sqsubseteq_P, s) \wedge S_{Q,i}(t, \sqsubseteq_Q, \prec\succ) \wedge s \neq \prec\succ \wedge \mathit{begin}\ s = t \\
& \qquad \wedge (\mathit{head}\ s \sqsupset_P (t, \{\!\|\!\}) \vee \mathit{head}\ s \notin \mathrm{items} \sqsubseteq_Q) \\
& \vee S_{P,i}(t, \sqsubseteq_P, \prec\succ) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s) \wedge s \neq \prec\succ \\
& \qquad \wedge \mathit{begin}\ s = t \wedge \mathit{head}\ s \not\sqsupset_P (\tau_P, \{\!\|\!\})
\end{aligned}
$$

$\triangle$

B.3.5 Parallel composition

Rule B.3.13 $(P \parallel Q)$

$$\frac{\begin{array}{ll}\forall i & P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_{P,i} \mid G_{P,i} \\ \forall i & Q\ \mathbf{sat}_{\rho}^{\geqslant q_i}\ S_{Q,i} \mid G_{Q,i} \\ \forall i & \begin{pmatrix} S_{P,i}(\tau, \sqsubseteq_P, s) \wedge G_{P,i}(\tau, \sqsubseteq_P, s) \\ \wedge\ S_{Q,i}(\tau, \sqsubseteq_Q, s) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s) \end{pmatrix} \Rightarrow \begin{pmatrix} S(\tau, \sqsubseteq_P \parallel \sqsubseteq_Q, s) \\ \wedge\ G(\tau, \sqsubseteq_P \parallel \sqsubseteq_Q, s) \end{pmatrix} \\ \forall i & G(\tau, \sqsubseteq_P \parallel \sqsubseteq_Q, s) \Rightarrow G_{P,i}(\tau, \sqsubseteq_P, s) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s) \end{array}}{P \parallel Q\ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i\, p_i q_i}\ S \mid G} \left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ \textit{disjoint} \end{array} \right]$$

where $\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\widehat{=}} S_{P,i}(\tau, \sqsubseteq_P, s) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s)$. $\triangle$

Rule B.3.14 $(P\ {}^{X}\!\parallel^{Y}\ Q)$

$$\frac{\begin{array}{ll}\forall i & P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_{P,i} \mid G_{P,i} \\ \forall i & Q\ \mathbf{sat}_{\rho}^{\geqslant q_i}\ S_{Q,i} \mid G_{Q,i} \\ \forall i & \begin{pmatrix} \Sigma s \subseteq X \cup Y \wedge S_{P,i}(\tau, \sqsubseteq_P, s \quad X) \wedge G_{P,i}(\tau, \sqsubseteq_P, s \quad X) \\ \wedge\ S_{Q,i}(\tau, \sqsubseteq_Q, s \quad Y) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, s \quad Y) \end{pmatrix} \Rightarrow \\ & \quad S(\tau, \sqsubseteq_P {}^{X}\!\parallel^{Y} \sqsubseteq_Q, s) \wedge G(\tau, \sqsubseteq_P {}^{X}\!\parallel^{Y} \sqsubseteq_Q, s) \\ \forall i & \begin{pmatrix} \Sigma s \subseteq X \cup Y \\ \wedge\ G(\tau, \sqsubseteq_P {}^{X}\!\parallel^{Y} \sqsubseteq_Q, s) \end{pmatrix} \Rightarrow \begin{pmatrix} G_{P,i}(\tau, \sqsubseteq_P, s \quad X) \\ \wedge\ G_{Q,i}(\tau, \sqsubseteq_Q, s \quad Y) \end{pmatrix} \end{array}}{P\ {}^{X}\!\parallel^{Y}\ Q\ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i\, p_i q_i}\ S \mid G} \left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ \textit{disjoint} \end{array} \right]$$

where $\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\widehat{=}} \Sigma s \subseteq X \cup Y \wedge S_{P,i}(\tau, \sqsubseteq_P, s \quad X) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, s \quad Y)$. $\triangle$

B.3.6 Interleaving

Rule B.3.15 $(P \longleftarrow Q)$

$$\frac{\begin{array}{ll}\forall i & P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_{P,i} \mid G_{P,i} \\ \forall i & Q\ \mathbf{sat}_{\rho}^{\geqslant q_i}\ S_{Q,i} \mid G_{Q,i} \\ \forall i & \begin{pmatrix} S_{P,i}(\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \wedge G_{P,i}(\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \\ \wedge\ S_{Q,i}(\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \wedge G_{Q,i}(\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \end{pmatrix} \Rightarrow \\ & \quad S(\tau, \sqsubseteq_P \longleftarrow \sqsubseteq_Q, s) \wedge G(\tau, \sqsubseteq_P \longleftarrow \sqsubseteq_Q, s) \\ \forall i & G(\tau, \sqsubseteq_P \longleftarrow \sqsubseteq_Q, s) \Rightarrow \begin{pmatrix} G_{P,i}(\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \\ \wedge\ G_{Q,i}(\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} s \end{pmatrix} \end{array}}{P \longleftarrow Q\ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i\, p_i q_i}\ S \mid G} \left[\begin{array}{c} \langle \hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \rangle \\ \textit{disjoint} \end{array} \right]$$

where $\hat{S}_i(\tau, \sqsubseteq_P, \sqsubseteq_Q, s) \mathrel{\widehat{=}} S_{P,i}(\tau, \sqsubseteq_P, \Uparrow_{\sqsubseteq_P, \sqsubseteq_Q} s) \wedge S_{Q,i}(\tau, \sqsubseteq_Q, \Downarrow_{\sqsubseteq_P, \sqsubseteq_Q} s)$. $\triangle$

B.3.7 Abstraction and renaming

Rule B.3.16 $(P \setminus X)$

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p} \ S_P \mid G_P \\ S_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{\setminus X} s) \wedge G_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{\setminus X} s) \Rightarrow S(\tau, \sqsubseteq \setminus X, s) \wedge G(\tau, \sqsubseteq \setminus X, s) \\ G(\tau, \sqsubseteq \setminus X, s) \Rightarrow G_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{\setminus X} s) \end{array}}{P \setminus X \ \mathbf{sat}_{\rho}^{\geqslant p} \ S \mid G}$$

$\triangle$

Rule B.3.17 $(f(P))$

$$\frac{\begin{array}{l} P \ \mathbf{sat}_{\rho}^{\geqslant p} \ S_P \mid G_P \\ S_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{f} s) \wedge G_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{f} s) \Rightarrow S(\tau, f\sqsubseteq, s) \wedge G(\tau, f\sqsubseteq, s) \\ G(\tau, f\sqsubseteq, s) \Rightarrow G_P(\tau, \sqsubseteq, \Uparrow_{\sqsubseteq}^{f} s) \end{array}}{f(P) \ \mathbf{sat}_{\rho}^{\geqslant p} \ S \mid G}$$

$\triangle$

B.3.8 Transfer operators

Rule B.3.18 $(P \ {}^{t} \ Q)$

$$\frac{\begin{array}{ll} \forall i & P \ \mathbf{sat}_{\rho}^{\geqslant p_i} \ S_{P,i} \mid G_{P,i} \wedge (\tau < t \vee \mathit{begin}\, s \quad t) \\ \forall i & P \ \mathbf{sat}_{\rho}^{\geqslant p'_i} \ S'_{P,i} \mid G'_{P,i} \wedge \tau = t \wedge s = \prec\succ \\ \forall i & Q \ \mathbf{sat}_{\rho}^{\geqslant q_i} \ S_{Q,i} \mid G_{Q,i} \\ \forall i & S_{P,i}(\tau, \sqsubseteq, s) \wedge G_{P,i}(\tau, \sqsubseteq, s) \wedge (\tau < t \vee \mathit{begin}\, s \quad t) \Rightarrow \\ & \qquad S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\ \forall i & S'_{P,i}(t, \sqsubseteq, \prec\succ) \wedge G'_{P,i}(t, \sqsubseteq, \prec\succ) \wedge t \quad \tau < t + \delta \Rightarrow \\ & \qquad S((t, \sqsubseteq, \prec\succ) \quad \mathrm{empty}_{(t,\tau]}) \wedge G((t, \sqsubseteq, \prec\succ) \quad \mathrm{empty}_{(t,\tau]}) \\ \forall i & \begin{pmatrix} S'_{P,i}(t, \sqsubseteq_P, \prec\succ) \wedge G'_{P,i}(t, \sqsubseteq_P, \prec\succ) \\ \wedge\ S_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \wedge G_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \end{pmatrix} \Rightarrow \\ & \qquad \begin{pmatrix} S((t, \sqsubseteq_P, \prec\succ) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \\ \wedge\ G((t, \sqsubseteq_P, \prec\succ) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \end{pmatrix} \\ \forall i & G(\tau, \sqsubseteq, s) \wedge (\tau < t \vee \mathit{begin}\, s \quad t) \Rightarrow G_{P,i}(\tau, \sqsubseteq, s) \\ \forall i & G((t, \sqsubseteq, \prec\succ) \quad \mathrm{empty}_{(t,\tau]}) \wedge t \quad \tau < t + \delta \Rightarrow G'_{P,i}(t, \sqsubseteq, \prec\succ) \\ \forall i & G((t, \sqsubseteq_P, \prec\succ) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \Rightarrow \\ & \qquad G'_{P,i}(t, \sqsubseteq_P, \prec\succ) \wedge G_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \end{array}}{P \ {}^{t} \ Q \ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i p_i} \ S \mid G} \begin{bmatrix} (\hat{S}_i(\tau, \sqsubseteq, s)) \ \mathit{disjoint} \\ (\hat{S}'_i(\tau, \sqsubseteq, s)) \ \mathit{disjoint} \\ \Sigma_i\, p'_i q_i \quad \Sigma_i\, p_i \end{bmatrix}$$

where i ranges over some set I and

$$\hat{S}_i(\tau, \sqsubseteq, s) \ \widehat{=} \ (\tau < t \vee \mathit{begin}\, s \quad t) \wedge S_{P,i}(\tau, \sqsubseteq, s) \wedge G_{P,i}(\tau, \sqsubseteq, s)$$
$$\hat{S}'_i(\tau, \sqsubseteq, s) \ \widehat{=} \ \tau = t \wedge s = \prec\succ \wedge S'_{P,i}(\tau, \sqsubseteq, s) \wedge G'_{P,i}(\tau, \sqsubseteq, s)$$

$\triangle$

Rule B.3.19 $(P \;_{t}\; Q)$

$$
\frac{\begin{array}{l}
\forall i \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_{P,i} \mid G_{P,i} \wedge \tau < t \\
\forall i \quad P\ \mathbf{sat}_{\rho}^{\geqslant p'_i}\ S'_{P,i} \mid G'_{P,i} \wedge \tau = t \\
\forall i \quad Q\ \mathbf{sat}_{\rho}^{\geqslant q_i}\ S_{Q,i} \mid G_{Q,i} \\
\forall i \quad S_{P,i}(\tau, \sqsubseteq, s) \wedge G_{P,i}(\tau, \sqsubseteq, s) \wedge \tau < t \Rightarrow S(\tau, \sqsubseteq, s) \wedge G(\tau, \sqsubseteq, s) \\
\forall i \quad S'_{P,i}(t, \sqsubseteq, s) \wedge G'_{P,i}(t, \sqsubseteq, s) \wedge t \quad \tau < t + \delta \Rightarrow \\
\qquad S((t, \sqsubseteq, s) \quad \mathrm{empty}_{(t,\tau]}) \wedge G((t, \sqsubseteq, s) \quad \mathrm{empty}_{(t,\tau]}) \\
\forall i \quad \begin{pmatrix} S'_{P,i}(t, \sqsubseteq_P, s_P) \wedge G'_{P,i}(t, \sqsubseteq_P, s_P) \\ \wedge\ S_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \wedge G_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q) \end{pmatrix} \Rightarrow \\
\qquad \begin{pmatrix} S((t, \sqsubseteq_P, s_P) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \\ \wedge\ G((t, \sqsubseteq_P, s_P) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \end{pmatrix} \\
\forall i \quad G(\tau, \sqsubseteq, s) \wedge \tau < t \Rightarrow G_{P,i}(\tau, \sqsubseteq, s) \\
\forall i \quad G((t, \sqsubseteq, s) \quad \mathrm{empty}_{(t,\tau]}) \wedge t \quad \tau < t + \delta \Rightarrow G'_{P,i}(t, \sqsubseteq, s) \\
\forall i \quad G((t, \sqsubseteq_P, s_P) \quad \mathrm{empty}_{(t,t+\delta)} \quad (\tau_Q, \sqsubseteq_Q, s_Q) + t + \delta) \Rightarrow \\
\qquad G'_{P,i}(t, \sqsubseteq_P, s_P) \wedge G_{Q,i}(\tau_Q, \sqsubseteq_Q, s_Q)
\end{array}}{P \;_{t}\; Q\ \mathbf{sat}_{\rho}^{\geqslant \Sigma_i p_i}\ S \mid G}
\begin{bmatrix} \langle \hat{S}_i(\tau, \sqsubseteq, s) \rangle \\ \quad disjoint \\ \langle \hat{S}'_i(\tau, \sqsubseteq, s) \rangle \\ \quad disjoint \\ \Sigma_i\, p'_i q_i \quad \Sigma_i\, p_i \end{bmatrix}
$$

where i ranges over some set I and

$$
\begin{aligned}
\hat{S}_i(\tau, \sqsubseteq, s) &\mathrel{\widehat{=}} \tau < t \wedge S_{P,i}(\tau, \sqsubseteq, s) \wedge G_{P,i}(\tau, \sqsubseteq, s) \\
\hat{S}'_i(\tau, \sqsubseteq, s) &\mathrel{\widehat{=}} \tau = t \wedge S'_{P,i}(\tau, \sqsubseteq, s) \wedge G'_{P,i}(\tau, \sqsubseteq, s)
\end{aligned}
$$

$\triangle$

Rule B.3.20 $(P \underset{e}{\nabla} Q)$

$$
\frac{\begin{array}{l}
P\ \mathbf{sat}_{\rho}^{\geqslant p}\ S_P \\
Q\ \mathbf{sat}_{\rho}^{\geqslant q}\ S_Q \\
S_P(\tau, \sqsubseteq, s) \wedge e \notin \Sigma s \Rightarrow S(\tau, \sqsubseteq \oplus e, s) \\
S_P(t, \sqsubseteq, s) \wedge e \notin \Sigma s \wedge t \quad \tau < t + \delta \Rightarrow \\
\qquad S((t, \sqsubseteq \oplus e, s \quad (t, e)) \quad \mathrm{empty}_{(t,\tau]}) \\
S_P(t, \sqsubseteq, s) \wedge e \notin \Sigma s \wedge S_Q(b_Q) \Rightarrow \\
\qquad S((t, \sqsubseteq \oplus e, s \quad (t, e)) \quad \mathrm{empty}_{(t,t+\delta)} \quad b_Q + t + \delta)
\end{array}}{P \underset{e}{\nabla} Q\ \mathbf{sat}_{\rho}^{\geqslant pq}\ S}
\begin{bmatrix} S_P \text{ is a safety} \\ \text{predicate} \end{bmatrix}
$$

$\triangle$

B.3.9 Recursion

Rule B.3.21 ($\mu X \quad P$, **conditional specifications**)

$$
\frac{(\forall i \quad X\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i \mid G_i) \ \Rightarrow\ (\forall i \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i \mid G_i)}{\forall i \quad \mu X \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i \mid G_i}
$$

$\triangle$

Rule B.3.22 ($\mu X \quad P$, **unconditional specifications**)

$$\frac{(\forall i \quad X\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i) \Rightarrow (\forall i \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i)}{\forall i \quad \mu X \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i} \left[\begin{array}{c} \forall i \quad \exists P : \mathcal{M}_{PTB} \\ P\ \mathbf{sat}_{\rho}\ S_i \end{array}\right]$$

$\triangle$

Rule B.3.23 ($\mu X \quad P$, **conditional specifications**)

$$\frac{\begin{array}{c}\left(\forall i \quad X\ \mathbf{sat}_{\rho}^{\geqslant p_i} \left(\begin{array}{l} S_i((\tau, \sqsubseteq, s) - \delta) \wedge begin\ s \quad \delta \\ \wedge \sqsubseteq \quad \delta = [0,\delta) \otimes \langle \{\!|\,|\!\} \rangle \end{array} \middle| \begin{array}{l} G_i((\tau, \sqsubseteq, s) - \delta) \wedge begin\ s \quad \delta \\ \wedge \sqsubseteq \quad \delta = [0,\delta) \otimes \langle \{\!|\,|\!\} \rangle \end{array}\right)\right) \Rightarrow \\ (\forall i \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i \mid G_i)\end{array}}{\forall i \quad \mu X \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i \mid G_i}$$

$\triangle$

Rule B.3.24 ($\mu X \quad P$, **unconditional specifications**)

$$\frac{\begin{array}{c}\left(\forall i \quad X\ \mathbf{sat}_{\rho}^{\geqslant p_i} \left(\begin{array}{l} S_i((\tau, \sqsubseteq, s) - \delta) \wedge begin\ s \quad \delta \\ \wedge \sqsubseteq \quad \delta = [0,\delta) \otimes \langle \{\!|\,|\!\} \rangle \end{array}\right)\right) \Rightarrow \\ (\forall i \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i)\end{array}}{\forall i \quad \mu X \quad P\ \mathbf{sat}_{\rho}^{\geqslant p_i}\ S_i} \left[\begin{array}{c} \forall i \quad \exists P : \mathcal{M}_{PTB} \\ P\ \mathbf{sat}_{\rho}\ S_i \end{array}\right]$$

$\triangle$

Rule B.3.25 ($\langle X_i = P_i \rangle_k$, **conditional specifications**)

$$\frac{\begin{array}{l}(\forall i : I\ ; j : J \quad X_i\ \mathbf{sat}_{\rho}^{\geqslant p_{i,j}}\ S_{i,j} \mid G_{i,j}) \Rightarrow \\ \quad (\forall i : I\ ; j : J \quad P_i\ \mathbf{sat}_{\rho}^{\geqslant p_{i,j}}\ S_{i,j} \mid G_{i,j})\end{array}}{\forall j : J \quad \langle X_i = P_i \rangle_k\ \mathbf{sat}_{\rho}^{\geqslant p_{k,j}}\ S_{k,j} \mid G_{k,j}}$$

$\triangle$

Rule B.3.26 ($\langle X_i = P_i \rangle_k$, **unconditional specifications**)

$$\frac{\begin{array}{l}(\forall i : I\ ; j : J \quad X_i\ \mathbf{sat}_{\rho}^{\geqslant p_{i,j}}\ S_{i,j}) \Rightarrow \\ \quad (\forall i : I\ ; j : J \quad P_i\ \mathbf{sat}_{\rho}^{\geqslant p_{i,j}}\ S_{i,j})\end{array}}{\forall j : J \quad \langle X_i = P_i \rangle_k\ \mathbf{sat}_{\rho}\ S_{k,j}} \left[\begin{array}{c} \forall i : I\ ; j : J \quad \exists P : \mathcal{M}_{PTB} \\ P\ \mathbf{sat}\ S_{i,j} \end{array}\right]$$

$\triangle$

Bibliography

[AH90] James Aspnes and Maurice Herlihy. Fast randomized consensus using shared memory. *Journal of Algorithms*, 11:441–461, 1990.

[BBK85] J. C. M. Baeten, J. A. Bergstra, and J. W. Klop. Syntax and defining eqnations for an interrupt mechanism in process algebra. Technical Report CS-R8503, Centre for Mathematics and Computer Science, Amsterdam, 1985.

[BHR84] S. D. Brookes, C. A. R. Hoare, and A. W. Roscoe. A theory of CSP. *Journal of the ACM*, 31(3):560–599, 1984.

[BW82] M. Broy and M. Wirsling. On the algebraic specification of finitary infinite communicating sequential processes. In D. Björner, editor, *Working Conference on Formal Description of Programming Concept II*, Amsterdam, 1982. North Holland.

[Cam89] J. Camilleri. Introducing a priority operator to CCS. Technical Report 157, Cambridge, 1989.

[CES83] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications: A practical approach. In *Proceedings of 10th ACM Symposium on Principles of Programming Languages*, pages 117–126, 1983.

[CH88] R. Cleaveland and M. Hennessy. Priorities in process algebras. In *Proc. 3rd Symposium on Logic in Computer Science, Edinburgh*, 1988.

[Chr90] Ivan Christoff. Testing equivalences and fully abstract models for probabilistic processes. In *Concur '90, LNCS 458*. Springer Verlag, 1990.

[Dav91] Jim Davies. *Specification and Proof in Real-Time Systems*. D. Phil thesis, Oxford University, 1991. Published as Oxford University Computing Labs, Technical Monograph PRG-93.

[DRRS93] J. Davies, G. M. Reed, A. W. Roscoe, and S. A. Schneider. *Real Time CSP*. Prentice Hall, 1993. Forthcoming.

[DS89a] Jim Davies and Steve Schneider. An introduction to Timed CSP. Technical Monograph PRG-75, Oxford University Computing Labs, 1989.

[DS89b] Jim Davies and Steve Schneider. Factorising proofs in Timed CSP. Technical Monograph PRG-75, Oxford University Computing Labs, 1989.

[DS90] Jim Davies and Steve Schneider. Waiting for Timed CSP. Technical Report PRG-TR-3-90, Oxford University Computing Labs, 1990.

[DS92a] Jim Davies and Steve Schneider. A brief history of Timed CSP. Technical Monograph 96, Oxford University Computing Labs, 1992.

[DS92b] Jim Davies and Steve Schneider. Using CSP to verify a timed protocol over a fair medium. In *CONCUR '92, LNCS 630*, 1992.

[FZHS92] M. Fang, H. S. M. Zedan, and C. J. Ho-Stuart. A theory for timed-probabilistic behaviours. Report YCS 175, University of York, Department of Computer Science, 1992.

[GJS90] A. Giacalone, C. Jou, and S. A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In *Proceedings of Working Conference on Programming Concepts and Methods, IFIP TC 2*, 1990.

[Han91] Hans A. Hansson. *Time and Probability in Formal Design of Distributed Systems*. PhD thesis, Swedish Institute of Computer Science, 1991. Published as SICS Dissertation Series, number 05.

[Her90] Ted Herman. Probabilistic self-stabilization. *Information Processing Letters*, 35(2):63–67, June 1990.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[HSZFH92] C. J. Ho-Stuart, H. S. M. Zedan, M. Fang, and C. M. Holt. PARTY: A process algebra with real-time from York. Report YCS 177, University of York, Department of Computer Science, 1992.

[JL91] Bengt Jonsson and Kim G. Larsen. Specification and refinement of probabilistic processes. In *Proc. LICS '91*, 1991.

[JS90] Chi-Chang Jou and Scott A. Smolka. Equivalences, congruences and complete axiomatizations for probabilistic processes. In *Concur '90, LNCS 458*, 1990.

[Low91a] Gavin Lowe. A probabilistic model of Timed CSP. D. Phil qualifying thesis, Oxford, 1991.

[Low91b] Gavin Lowe. Prioritized and probabilistic models of Timed CSP. Technical Report PRG-TR-24-91, Oxford University Computing Labs, 1991.

[Low92a] Gavin Lowe. Some extensions to the Probabilistic, Biased Model of Timed CSP. Technical Report PRG-TR-9-92, Oxford University Computing Labs, 1992.

[Low92b] Gavin Lowe. Relating the Prioritized Model of Timed CSP to the Timed Failures Model. Technical Report PRG-TR-18-92, Oxford University Computing Labs, 1992.

[Low92c] Gavin Lowe. Specification and proof in probabilistic, prioritized, Timed CSP. Technical Report PRG-TR-23-92, Oxford University Computing Labs, 1992.

[LS92] Kim G. Larsen and Arne Skou. Compositional verification of probabilistic processes. In *Concur '92, LNCS 630*, 1992.

[Mil83] Robin Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.

[Mil89] Robin Milner. *Communication and Concurrency.* Prentice Hall International, 1989.

[Mor90] Carroll Morgan. *Programming from Specifications.* Prentice Hall, 1990.

[OH83] E. R. Olderog and C. A. R. Hoare. Specification-oriented semantics for communicating processes. In J. Diaz, editor, *10th ICALP, LNCS 154*, pages 561–572, 1983.

[PS88] K. Paliwoda and J. W. Sanders. The sliding window protocol in CSP. Technical Monograph PRG-66, Oxford University Computing Labs, 1988.

[PZ86] A. Pnueli and L. Zuck. Verification of multiprocess probabilistic protocols. *Distributed Computing*, 1(1):53–72, 1986.

[Ree88] G. M. Reed. *A Uniform Mathematical Theory for Real-Time Distributed Computing.* D. Phil thesis, Oxford University Computing Labs, 1988.

[Ros82] A. W. Roscoe. *A Mathematical Theory of Communicating Processes.* D. Phil thesis, Oxford, 1982.

[RR86] G. M. Reed and A. W. Roscoe. A timed model for CSP. In *Proceedings of ICALP86, LNCS 226; Theoretical Computer Science 58*, pages 314–323. Springer Verlag, 1986.

[RR87] G. M. Reed and A. W. Roscoe. Metric spaces as models for real-time concurrency. In *Proceedings of the Third Workshop on the Mathematical Foundations of Programming Semantics, LNCS 298*, pages 331–343. Springer Verlag, 1987.

[Sch90] Steve Schneider. *Correctness and Communication in Real Time Systems.* D. Phil thesis, Oxford University, 1990. Published as Oxford University Computing Labs, Technical Monograph PRG-88.

[Sco92] Brian Scott. Denotational semantics for occam 2. D. Phil qualifying thesis, Oxford, 1992.

[Sei92] Karen Seidel. *Probabilistic Communicating Processes.* D. Phil thesis, Oxford University, 1992.

[SS90] Scott A. Smolka and Bernhard Steffen. Priority as extremal probability. In *Concur '90, LNCS 458.* Springer Verlag, 1990.

[Sut75] W. A. Sutherland. *Introduction to Metric and Topological Spaces.* Oxford University Press, 1975.

[Tof90] Chris Tofts. A synchronous calculus of relative frequency. In *Concur '90, LNCS 458.* Springer Verlag, 1990.

[vGSST90] R. J. van Glabbeek, S. A. Smolka, B. Steffen, and C. Tofts. Reactive, generative and stratified models of probabilistic processes. In *IEEE Symposium on Logic in Computer Science*, 1990.

Index of Notation

Syntax

Semantics

Timed Failures Model

Operations on timed failures

The Prioritized Model

s	a timed trace	30	TT	all timed traces	30
$\langle\rangle$	the empty trace	30	OFF	all offers	30
χ, ψ	bags of events	30	ACT	all actions	63
α, β	actions	63	$OFFREL$	all offer relations	30
v, w	offers	30	BEH	all prioritized behaviours	34
$\sqsubseteq$	an offer relation; offered less strongly than	30	$EOFF$	all environmental offers	38
$\sqsubset$	offered strictly less strongly than	31	$\mathcal{S}_{TB}$	all sets of prioritized behaviours	39
$\sqsupseteq$	offered stronger than	31	$\mathcal{M}_{TB}$	Timed Prioritized Model	39
$\sqsupset$	offered strictly stronger than	31	ENV	variable bindings	40
τ	end time	31	$\mathcal{A}_{BT}$	prioritized behaviours	40
Ω	an environmental offer	38			

The Deterministic Model

$\mathcal{M}_{DTB}$	Deterministic, Timed Model using Biases	65	$\mathcal{A}_{DT}$	deterministic behaviours	65

The Probabilistic Model

$\mathcal{PF}_{TB}$	probability functions on timed biased behaviours	68	$\mathcal{A}_{PBT}$	behaviours of probabilistic process	71
$\mathcal{PP}_{TB}$	probabilistic pairs using timed biased behaviours	68	$\mathcal{P}_{PBT}$	probability functions on prioritized behaviours	71
$\mathcal{M}_{PTB}$	Probabilistic, Timed Model using Biases	70	$\mathcal{F}_{PBT}$	probabilistic pairs	71
			fillout	extend partial function	72

Operations on prioritized behaviours

Specification

Timed Failures Model

sat	satisfies	17
sat_ρ	satisfies in variable binding ρ	17
at	performance of events	18
ref	refusal of events	19
beyond	time at least	19
live	willing to perform events	19
from	all times after	20
first	first timed event	20
last	last timed event	20
after	after	20
before	before	20
during	during interval	20
time of	time of timed event	20
name of	name of timed event	20
alphabet	set of all events performed	21
count	number of events performed	21
open	offered by environment	21
closed	not offered by environment	22
internal	internal events	22

Prioritized and Probabilistic Models

sat	satisfies	80
sat_ρ	satisfies in variable binding ρ	80
$^{\Omega,\rho}_{P}\ S$	probability of satisfying S	134
$\mathsf{sat}^{\geqslant p}$	satisfies with probability p	134
$\mathsf{sat}^{\geqslant p}_{\rho}$	satisfies with probability p in variable binding ρ	134
$S \mid G$	conditional specification	134
disjoint	specifications disjoint	139
at	performance of events	83
ref	refusal of events	84
beyond	time at least	84
offered	event offered by process	84
live	willing to perform events	85
from	all times after	85
separate	two events not offered at the same time	86
preferred to	priorities on actions	86
first	first timed event	87
last	last timed event	87
after	after	87
before	before	87
during	during interval	87
time of	time of timed event	87
name of	name of timed event	87
alphabet	set of all events performed	87
count	number of events performed	87
open	offered by environment	88
closed	not offered by environment	88
internal	internal events	88
accessible	action offered by environment	89

Abstraction

$\varphi_P^{(B)}$	unprobabilizing syntactic abstraction	81
$\varphi_D^{(B)}$	nondeterminizing syntactic abstraction	83
φ_B	unprioritizing syntactic abstraction	107
$\sim$	equivalence of traces	108
ref	total refusal set relating to prioritized behaviour	109
closure	left hand closure of refusal sets	109
$\simeq$	compatibility of timed failure with prioritized behaviour	109
$\theta_P^{(B)}$	unprobabilizing semantic abstraction	81
$\theta_D^{(B)}$	nondeterminizing semantic abstraction	83
θ_B	unprioritizing semantic abstraction	110
$\mathcal{A}_{PT}$	timed failures of prioritized process	110
Θ_B	abstraction mapping for specifications	120