The existence of a normal form permits other ~~op~~ new operators to be ~~int~~ defined in the algebra by means of structural induction on normal forms.

if Π has no unit then $\{\}$ is disallowed.

For example let ~~a.~~ f be a function on atoms. This may be "lifted" to (Π-normal forms by defining

$$f^{\Pi}(N) \stackrel{df}{=} \Pi\{ft \mid t \in N\} \quad (*)$$

This clearly distributes through Π, and f Often, the superscript is omitted. A binary operator $\circ$ may be similarly lifted

$$N \circ M = \Pi\{t \circ u \mid t \in N \,\&\, u \in N\} \quad (\circ)$$

If c is an ato

Lifting is clearly a functor that distributes through Π

$$(f;g)^{\Pi} = f^{\Pi};g^{\Pi}$$

$$f^{\Pi}(N \Pi M) = f^{\Pi}(N) \,\Pi\, f^{\Pi}(M)$$

Our next step is to take the same step again by introducing a third associative operator $\wedge$, with zero $\leftrightarrow$, defined on atoms. It distributes over $\sqcup$. We postulate it to be associative, commutative and idempotent, and on atoms. We want it to be the innermost operator on normal forms. We therefore lift it in the usual way

$$N \wedge M = \bigsqcap \{ t \wedge u \mid t \in N \,\&\, u \in M \}$$

$$t \wedge u = \bigsqcup \{ a \wedge b \mid a \in t \,\&\, b \in u \}$$

This formula shows that $\sqcup$ could equally well be defined as a lifted (set-) union on normal forms. A lifted (set-) intersection could be defined in the same way

$$N \,||\, M = \sqcap \{ t \cup u \mid t \in N \,\&\, u \in M \}.$$

This operator is also associative and commutative, and distributes through $\sqcap$. But it is not idempotent and it does not distribute through $\sqcup$. For example

$$(\{\{a\},\{b\}\} \,||\, \{\{a\},\{b\}\}) = \{\{a\} \cap \{a\}, \{a\} \cap \{b\}, \{b\} \cap \{b\}\}$$
$$= \{\{a\}, \{b\}, \{\}\}$$

Lifting a function F through our $(\Pi, \mathcal{H})$ normal form is a multistage process.

The definition can be slightly generalised

$$f(N) = \Pi \left\{ F_c^{\mathcal{H}}(t) \mid t \in N \right\}$$

The definition of lifting can be slightly generalised to introduce a constant c into the normal form

$$f_c^{\Pi}(N) = c, \Pi\{f_t \mid t \in N\}$$

Extended summary.

A prototypical normal form contains two operators, an outer operator $\sqcap$ and an inner operator $\sqcup$, where $\sqcup$ distributes through $\sqcap$. We assume both operators are associative, commutative, and idempotent, so that a normal form is essentially a family of sets $(d, e, \ldots)$ of more primitive terms $(s, t, \ldots)$. An arbitrary term using $\sqcap$ and $\sqcup$ in arbitrary nesting may be reduced to normal form

$$P \sqcap Q = P \cup Q$$

$$P \sqcup Q = \{d \cup e \mid d \in P \;\&\; e \in Q\}$$

Unfortunately not idempotent.

Note that these operators preserve finitude and nonemptiness of their operands. That will be an important property of all operators of a calculus.

When two normal forms are unequal, we would like to display a single primitive term on which they disagree. This would be a term included in a set of one of them is included in the other for it could be a term excluded from a set of one of the apparently distinct forms. or

$$\{\{s\},\{s,t,u\}\} \quad \text{and} \quad \{\{s\},\{s,t\},\{s,u\},\{s,t,u\}\}$$

Each term possible for one of them is possible for the other. Each term excluded from a set in one of them is excluded from a set of the other. We must therefore identify these two normal forms, by choosing the largest of them as canonical. A canonical normal form is saturated, in that it satisfies (saturation) a convex closure condition

$$d, e \in P \;\&\; d \subseteq f \subseteq e \cup d \Rightarrow f \in P.$$

The $\sqcup$ operator preserves saturation, but we need to resaturate the result of the $\sqcap$ operation

$$P \sqcap Q = \{ f \mid \exists d, e \in P \cup Q,\ d \subseteq f \subseteq e \cup d \}$$

The use of this operator in the normal form relaxes the obligation We will see later how different closure conditions to classify define a family of related process calculi by the different closure conditions that they satisfy.

As a result of saturation we get the idempotence law $P \sqcup P = P$ of P and $P \sqcap (Q \sqcup R) = (P \sqcap Q) \sqcup (P \sqcap R)$

In the case of a process algebra, the primitive terms (called traces) are sequences of events. We also impose a closure condition on sets of traces

$$s^\frown t \in d \Rightarrow s \in d.$$

To abstract from the times at which precise observation is made,

As before the sets must be non-empty, i.e., they all contain the empty trace. Fortunately, the operators $\sqcap$ and $\sqcup$ both preserve this property, and so does a new operator $\|$, defined on normal forms by the distribution law

$$P \| Q = \{ d \frown e \mid d \in P \ \& \ e \in Q \}$$

We can also lift the prefix operator $a.t$ on traces t to processes

$$a.P = \{ a.d \mid d \in P \}$$

$$a.d = \{ a.t \mid t \in d \} \cup \{ \langle\rangle \}$$

This can be used as an "expansion law" to eliminate $\|$ from normal forms.

The operators π, $\sqcup$, a., and $\|$ have been selected with the single objective of simplifying the normal forms, and with no thought of interpretation, and even less of efficient implementation in a process calculus. In fact a.P is easily implemented by a process that engages in the event a, and then behaves like P. The operator π is also easy, when interpreted as demonic non-determinism (internal choice): just implement either of the operands. $\|$ and $\sqcup$ are implementable by executing both operands in parallel. In the case of $\|$, an event occurs only when both operands agree on it; and in the case of $\sqcup$, the environment is offered the choice of events which either operand offers. Unfortunately $\sqcup$ is expensive to implement, because the parallel execution of both operands gives a result and $\|$ is pretty useless as well. A practical process calculus must offer more efficient and more useful operators that could be equally well obtained by only one of them, if only we knew which.

A more efficient version of $\sqcup$ is the external choice, $\square$ in CSP, + in CCS.

$$P \square Q = \bigcup \{ d \sqcup e \mid d \in P \ \& \ e \in Q \}$$

where $d \sqcup e = \{ f \mid f_0 = d_0 \cup e_0$
$\& \ \forall a \in f_0 \ f/a = d/a, \vee f/a = e/a \}$

Parallelism can be made more efficient by priming each of its operands to accept but ignore certain of the events in which the other engages. Let E be a set of events not occurring in P. Then

$$P_{+E} = \{ d_{+E} \mid d \in P \}$$

$$d_{+E} = \{ s' \mid s \setminus E \in d \}$$

The par

The model developed so far gives a synchronous calculus very like CSP. ~~We~~ To "introduce" asynchrony into the model, ~~we define addition~~ we actually restrict the language by additional closure conditions, resulting from the fact that ~~all or some~~ some (or all) of the communication channels are buffered. These conditions are based on the Mazurkiewicz equivalences on traces. First, we need to distinguish certain of the event names from as inputs and others as outputs. Because input is buffered, an input can always appear earlier than stated; and an output can always appear later; we define an order relation $\rightarrow$

$t \rightarrow t'$ states that t' is formed from t by moving some of its inputs leftward and/or some of its outputs rightward. The relevant closure condition on traces is

if $t \rightarrow t'$ and $t \in d$ then $t' \in d$

Note that events that are not designated as either input or output remain synchronous.

Start with the simplest kind of algebra, with only one binary operator $\sqcap$, which is associative, commutative and idempotent. There are many such operators —

and, or, intersection, union, left, right, delta —

where $x \text{ left } y = x$

and $x \text{ delta } y = x$ if $x = y$, otherwise $\bot$

The normal form is just a list of distinct atoms (terms, each of which is an a constant or variable). It is "understood" that the ordering of the atoms is irrelevant: two normal forms are equal if they contain the same terms. Interesting variations in the algebra are:

(1) If the operator is not idempotent, (e.g. $+$, $\times$) the normal form is considered as a bag.

(2) If the operator is neither idempotent nor commutative, the normal form is a sequence

(3) If the operator has a unit $\top$, (e.g. 0 for $+$, 1 for $\times$) it is omitted from the normal form. (4) If it has a zero $\bot$, (e.g. 0 for $\times$) that is the only normal form that contains it. all other atoms are omitted from the term that contains it

true for and

true for or

More interesting algebras have two operators ($\sqcap$ and $\sqcup$) one of which ($\sqcup$) distributes through the other ($\sqcap$). Examples are $(+, \times)$, (and, or), (left, union). Now the normal form is a list of terms connected by $\sqcap$, where each term is a normal form for $\sqcup$. This is written

$$((a \sqcup b \sqcup \ldots) \sqcap (a' \sqcup b' \sqcup \ldots) \sqcap \ldots$$

which is formally abbreviated to

$$\prod_i \{\{\bigsqcup_j a \mid a \in t\} \mid t \in N\}$$

where N is a family of sets of atoms representing the normal form.

The way of proving that this is a normal form is to show that two normal forms connected by each operator can be reduced again to normal form

where $r_{i''j''} = p_{ij''}$ if $i'' < n$

$= q_{i''+n, j''}$ if $i'' \geq n$

$m''_{i''} = m_{i''}$ if i''

This In the case of $\sqcap$, this is done by appending the lists of outermost terms, and eliminating duplicates. In the case of $\sqcup$, it should be fully distributed through $\sqcap$, the resulting terms are reduced to normal form using the rules for $\sqcup$, and finally duplicate terms are eliminated. These rules are expressible in terms of families of sets

$$N \sqcap M \overset{\text{def}}{=} N \cup M$$

$$N \sqcup M = \{n \sqcup m \mid n \in N \,\&\, m \in M\}$$

Now let us introduce one more algebraic law, one that distributes the second operator through the first. Examples of pairs of operators enjoying such mutual ~~exclusion~~ distribution are (and, or), (left, union), (right, union). Clearly, the choice of outermost operator in the normal form is arbitrary; we will choose $\sqcap$. The effect of a new law is to equate normal forms that previously looked distinct. For example

$$p \sqcap q \sqcap (p \sqcup q) = (p \sqcap q \sqcap p) \sqcup (p \sqcap q \sqcap q) \qquad (\sqcap \text{ dist } \sqcup)$$

$$= p \sqcap q \qquad \text{symmetry, idempotence.}$$

$$p \sqcap (p \sqcup q \sqcup r) = (p \sqcap p) \sqcup (p \sqcap q) \sqcup (p \sqcap r) \qquad \sqcap \text{dist} \sqcup$$

$$= p \sqcap (p \sqcup q) \sqcap (p \sqcup r) \sqcap \qquad \sqcup \text{dist} \sqcap \text{ etc.}$$

$$(p \sqcup q \sqcup r)$$

In each case, both sides of the resulting equation are unequal normal forms.

A solution to this problem is to declare that one of each such pair is abnormal. It is convenient in theory (but not in practice) to reject the smaller of the two. The remaining normal forms are all saturated in the following sense.

(1) If the form contains terms p and q, it also contains $p \cup q$

(2) If the form contains terms p and $p \cup q \cup r$, it also contains $p \cup q$

More formally, we define

$$sat(N) = \{ t \mid \exists s, u \in N \; s \subseteq t \subseteq N \}$$

Luca
Silvano

Normal forms for synchronous and asynchronous process calculi.

Discussion paper between Tony Hoare and Cedric Fournet.

Revised 1999-12-09, and circulated for comment.

Symbolic calculation is a useful skill for engineers engaged in the design and validation of advanced technological products. The steps of the calculation are justified by algebraic laws, proved by mathematicians within the relevant scientific theory. A calculus is called complete if there are enough laws to prove every equation that is true in the theory. Preferably, the proofs should be automated, for example by reduction of both sides of an equation to some kind of normal form. Inequality of normal forms should be easily detectable, preferably by display of a counter-example.

Process algebra is the branch of mathematics that has been developed to analyse, predict, and control the behaviour of networks of interacting computers and programs. It deals with issues of security, robustness and performance. Its great achievement has been to apply the same mathematical abstractions at all levels of scale, from instruction pipelines in a single microprocessor through multiprogrammed personal computers, clusters and networks, right up to the entire world wide web. At the lower end of the scale, communication is usually synchronised, with input of a message occurring instantaneously with its output. Any overhead involved is minimal. On a larger scale, ~~messages are buffered in flight~~ between output and input. ~~Delays are inescapable~~, and overheads can be significant. Different branches of process algebra have been developed to deal with the synchronous and asynchronous cases.

there is an inescapable delay between

?

no need.

On occasion, there is a clear need for a calculus that combines both these modes of interaction. For example, asynchronous message passing is generally implemented on synchronous hardware; conversely, control signals, atomic actions and checkpoints have to be implemented on asynchronous networks as if they were instantaneous. Furthermore, we may want to transform programs and protocols between these two modes, to avoid latency and overhead by taking advantage of the characteristics of the architecture on which the programs are loaded.

(refinement)

We propose to recapitulate some known normal forms for synchronous and asynchronous process calculi [Hennessy, de Nicola, Brookes, Roscoe, Bergstra, Baeten], and suggest how they may be combined.

Mazurkiewicz ?

(asynchronous pi)

The known disadvantages of normal forms are

1. They involve taking limits of an infinite series of approximations, e.g., the Taylor expansion of an infinitely differentiable function. However, many of the symbolic calculations can be performed on the finite approximations, with only a single induction at the end.
   passage to the limit
2. They may use notations outside the limits of the familiar language used to state the problem, e.g., complex exponents in the Fourier expansion of a real periodic function. However, such notations can suggest concepts that turn out to be independently interesting and useful. for specification and for reasoning about designs

) reduction?

) SPEC ⊗ PRG.

suppose also $\vee$ distributes through $\cup$.

Then $s \vee t \vee (s \cup t) = s \vee t \vee s \cup s \vee t \vee t$ dist $\vee$ thru $\cup$

$= s \vee t$ assoc comm idem

$s \vee (s \cup t \cup u) = (s \vee s) \cup (s \vee t) \cup (s \vee u)$ dist $\vee$ thru $\cup$

$= s \vee (s \cup t) \vee (s \cup u) \vee (s \cup t \cup u)$ $\cup$ thru $\vee$

in general ~~$N = sat(N)$~~ $\bigvee\{U_s \mid s \in N\} = \bigvee\{U_s \mid s \in sat(N)\}$

$norm(N) = norm(sat(N))$

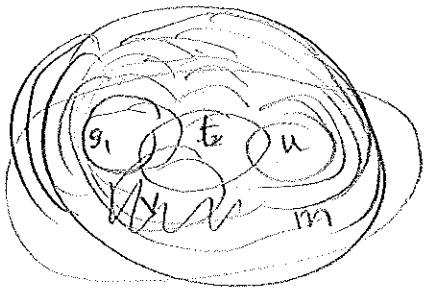
where $sat(N) = \bigvee\{U_b \mid \exists t \in N\ t \subseteq b \subseteq \bigcup N\}$

the one that satisfies $N = sat\,N$

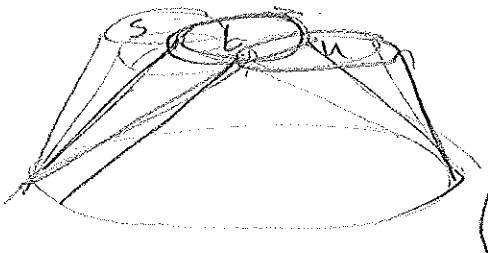
normal form is always saturated. or a pair

$pos(m, N)$ $\bigcup N \cup U_m \cup \bigvee\{U_s \mid s \in N\} \vee U_m \cup \bigcup\{U_s \mid s \in N\}$

where $\forall s \in N\ \neg s \subseteq t \,\&\, m \cap s = \emptyset$



$= (m, \{s, t, u, v\})$



other way

$(\{d\}, \{\{a, b\}, \{b, c\}\})$

$(a \wedge b) \vee (b \wedge c) \vee (c \wedge a) \vee (a \wedge b \wedge c \wedge d)$

normal forms.

One operator ϕ - union

may be associative

commutative

idempotent

unity false

zero true

normal form

ignore brackets

sort into fixed order

ignore duplicates

omit false

omit everything else

$a(bc)$ $a.(b.c)$ $(c.((a.a).b)).t.f$

$a.$

$c.a.a.b.t.f$

$a.a.b.b.f.t$

$a.b.b.f.t$

$a.b.c.t$

t

$\cup$ $\cap$

Two such operators, say $\vee$ and $\wedge$.

$\cup$ distributes through $\vee$

Normal form. is a family of sets where N is a set of sets

$\vee$ outermost

$$\bigvee \{ \bigwedge s \mid s \in N \}$$

proof $norm(N)$ of atoms

$NF \vee MF =$

$$\bigvee \{ \bigcup s \mid s \in N \} \vee \bigvee \{ \bigcup s \mid s \in M \} = \bigvee \{ \bigcup s \mid s \in M \cup N \}$$

$$\bigvee \{ \bigcup s \mid s \in N \} \cup \bigvee \{ \bigcup t \mid t \in M \} = \bigvee \{ \bigcup (s \cup t) \mid s \in N \,\&\, t \in M \}$$