# Calculating axiomatic semantics from program equations by means of functional predicate calculus

**(Some initial results of recent work — not for dissemination)**

Raymond Boute INTEC — Ghent University 2004/02

## Overview





# 0 Motivation

## 0.0 General

- Educational: axiomatic semantics (Hoare, Dijkstra) nonintuitive, "opaque"
- Research: further unification of mathematical methods for continuous and discrete systems (ongoing work)

## 0.1 Specific

- Justification of axiomatic semantics usually detours via denotational semantics e.g. Mike Gordon, Bertrand Meyer, Glynn Winskel
- Confusing terminology: what looks like *propositions* is often called *predicates*
- Correct semantics for assignment seems "backwards" (as observed by Gordon) Certain "forward" semantics is also correct (reinforces the "mystery"), e.g., $\{v = d\}\ v := e\ \{v = e[^{v}_{d}\}$ provided $v \notin \varphi\, d$

# 1 Calculating the "axioms" for assignment from equations

## 1.0 Principle

a. Basic ideas

- If pre- and postcondition *look* like propositions, *treat* them as such
- *Derive* axiomatic semantics from basic *program equations*. Convention: for program variable $v$, new vars: $\text{`}v$ before command, $v'$ after command.
  – Antecondition $a$ becomes $a[^{v}_{\text{`}v}$ and postcondition $p$ becomes $p[^{v}_{v'}$
  – Substitution and change of variables are familiar in engineering math
  – Consider axiomatic semantics just as "economy in variable use"
- Advantages of the approach:
  – Expressivity: direct formalization of intuitive program behaviour
  – Calculationally: all becomes predicate calculus (no "special" logics)

b. Convention

- Mnemonic symbols, even for bound variables (as in physics, applied math)
- We prefer "ante" over "pre" (better preposition, leads to distinct letters)



c. Expressing Floyd-Hoare semantics in terms of a program equation

- Side issue: assume $v$ to be of type $V$ (as specified by the declarations)
- Intuitive understanding of behaviour of assignment: equation $\boxed{v' = e[^{v}_{v'}}$
- Use in formalizing intuitive understanding of Floyd-Hoare semantics:
  – About $\text{`}v$ and $v'$ we know $a[^{v}_{\text{`}v}$ and $v' = e[^{v}_{\text{`}v}$ (no less, no more).
  – Hence any assertion about $v'$ must be implied by it, in particular $p[^{v}_{v'}$. Formally: $a[^{v}_{\text{`}v} \wedge v' = e[^{v}_{\text{`}v} \Rightarrow p[^{v}_{v'}$ (implicitly quantified over $\text{`}v$ and $v'$).

$$\boxed{\{a\}\ v := e\ \{p\} \ \equiv\ \forall\, \text{`}v : V \,.\, \forall\, v' : V \,.\, a[^{v}_{\text{`}v} \wedge v' = e[^{v}_{\text{`}v} \Rightarrow p[^{v}_{v'} \qquad (0)}$$

No detour via denotational semantics; assertions remain propositions.

- Example: assume `x` declared as integer. Then, by the preceding definition,

$$\{\mathtt{x} > 27\}\ \mathtt{x}\ \mathtt{:=}\ \mathtt{x{+}3}\ \{\mathtt{x} > 30\}$$
$$\equiv$$
$$\forall\, \text{`}\mathtt{x} : \mathbb{Z} \,.\, \forall\, \mathtt{x}' : \mathbb{Z} \,.\, \text{`}\mathtt{x} > 27 \wedge \mathtt{x}' = \text{`}\mathtt{x} + 3 \Rightarrow \mathtt{x}' > 30$$

The latter expression evaluates to 1 (or $\mathrm{T}$ if one prefers) by calculation.

## 1.1 Calculating the weakest antecondition

- Calculation (assuming type correctness checked, viz., $\forall\, \backprime v : V \,.\, e[^v_{\backprime v} \in V$)

$$
\begin{array}{lll}
\{a\}\ v := e\ \{p\} & \equiv & \langle \text{Definit. (0)} \rangle \quad \forall\, \backprime v : V \,.\, \forall\, v' : V \,.\, a[^v_{\backprime v} \wedge v' = e[^v_{\backprime v} \Rightarrow p[^v_{v'} \\
& \equiv & \langle \text{Shunting} \rangle \quad \forall\, \backprime v : V \,.\, \forall\, v' : V \,.\, a[^v_{\backprime v} \Rightarrow v' = e[^v_{\backprime v} \Rightarrow p[^v_{v'} \\
& \equiv & \langle \text{Rdist } \forall/\Rightarrow \rangle \quad \forall\, \backprime v : V \,.\, a[^v_{\backprime v} \Rightarrow \forall\, v' : V \,.\, v' = e[^v_{\backprime v} \Rightarrow p[^v_{v'} \\
& \equiv & \langle \text{One-pt. rule} \rangle \quad \forall\, \backprime v : V \,.\, a[^v_{\backprime v} \Rightarrow e[^v_{\backprime v} \in V \Rightarrow p[^v_{e[^v_{\backprime v}} \\
& \equiv & \langle \text{Assumption} \rangle \quad \forall\, \backprime v : V \,.\, a[^v_{\backprime v} \Rightarrow p[^v_{e[^v_{\backprime v}} \\
& \equiv & \langle \text{Change vars.} \rangle \quad \forall\, v : V \,.\, a \Rightarrow p[^v_e
\end{array}
$$

- This proves the $\boxed{\text{Theorem: } \{a\}\ v := e\ \{p\} \ \equiv\ \forall\, v : V \,.\, a \Rightarrow p[^v_e}$. Hence
  - $p[^v_e$ is at most as strong as any antecondition $a$.
  - $p[^v_e$ is itself an antecondition since $\{p[^v_e\}\ v := e\ \{p\} \ \equiv\ \forall\, v : V \,.\, p[^v_e \Rightarrow p[^v_e$
- Therefore $\boxed{\text{wa } [\![ v := e ]\!]\, p \ \equiv\ p[^v_e \qquad (1)}$



## 1.2 Calculating the strongest postcondition

- Calculation

$$
\begin{array}{lll}
\{a\}\ v := e\ \{p\} & \equiv & \langle \text{Definit. (0)} \rangle \quad \forall\, \backprime v : V \,.\, \forall\, v' : V \,.\, a[^v_{\backprime v} \wedge v' = e[^v_{\backprime v} \Rightarrow p[^v_{v'} \\
& \equiv & \langle \text{Swap } \forall/\forall \rangle \quad \forall\, v' : V \,.\, \forall\, \backprime v : V \,.\, a[^v_{\backprime v} \wedge v' = e[^v_{\backprime v} \Rightarrow p[^v_{v'} \\
& \equiv & \langle \text{Ldist } \forall/\Rightarrow \rangle \quad \forall\, v' : V \,.\, \exists\, (\backprime v : V \,.\, a[^v_{\backprime v} \wedge v' = e[^v_{\backprime v}) \Rightarrow p[^v_{v'} \\
& \equiv & \langle \text{Change var} \rangle \quad \forall\, v : V \,.\, \exists\, (\backprime v : V \,.\, a[^v_{\backprime v} \wedge v = e[^v_{\backprime v}) \Rightarrow p
\end{array}
$$

- Hence $\boxed{\text{Theorem: } \{a\}\ v := e\ \{p\} \ \equiv\ \forall\, v : V \,.\, \exists\, (u : V \,.\, a[^v_u \wedge v = e^v_u) \Rightarrow p}$, so
  - $\exists\, (u : V \,.\, a[^v_u \wedge v = e^v_u)$ is at least as strong as any postcondition $p$.
  - $\exists\, (u : V \,.\, a[^v_u \wedge v = e^v_u)$ is itself a postcondition.
- Therefore $\boxed{\text{sp } [\![ v := e ]\!]\, a \ \equiv\ \exists\, (u : V \,.\, a[^v_u \wedge v = e^v_u) \qquad (2)}$

### 1.3 A few interesting excursions / illustrations

a. Justifying the "forward" rule $\boxed{\{v = d\}\ v := e\ \{v = e[^v_d\}\quad \text{provided}\quad v \notin \varphi\, d}$

$$
\begin{array}{lll}
\{v = d\}\ v := e\ \{v = e[^v_d\} & & \\
\quad \equiv & \langle \text{Definit. (0)} \rangle & \forall\, `v : V \,.\, \forall\, v' : V \,.\, `v = d[^v_{`v} \wedge v' = e[^v_{`v} \Rightarrow v' = e[^v_{d[^v_{v'}} \\
\quad \equiv & \langle v \notin \varphi\, d \rangle & \forall\, `v : V \,.\, \forall\, v' : V \,.\, `v = d \wedge\ v' = e[^v_{`v} \Rightarrow v' = e[^v_d \\
\quad \equiv & \langle \text{Leibniz, bis} \rangle & \forall\, `v : V \,.\, \forall\, v' : V \,.\, `v = d \wedge\ v' = e[^v_{`v} \Rightarrow e[^v_{`v} = e[^v_{`v} \\
\quad \equiv & \langle \text{Reflex. } = \rangle & 1 \qquad (\text{or } \mathrm{T} \text{ if one prefers})
\end{array}
$$

b. Bouncing ante- and postconditions  Letting $c := [\![ v := e ]\!]$, calculation yields

$$
\begin{array}{l}
\mathsf{sp}\, c\, (\mathsf{wa}\, c\, p) \ \equiv\ p \wedge \exists\, u : V \,.\, v = e[^v_u \\
\mathsf{wa}\, c\, (\mathsf{sp}\, c\, a) \ \equiv\ \exists\, u : V \,.\, a[^v_u \wedge e = e^v_u
\end{array}
$$

E.g., $c :=$ 'y $:=$ y$^2 + 7$' and $p :=$ 'y $> 11$' and $q :=$ 'y $< 7$' and $a :=$ 'y $> 2$'

- sp $c$ (wa $c\, p$) $\equiv$ y $> 11 \wedge \exists\, x : \mathbb{Z} \,.$ y $= x^2 + 7$ (simplifies to y $> 11$)
- sp $c$ (wa $c\, q$) $\equiv$ y $< 7 \wedge \exists\, x : \mathbb{Z} \,.$ y $= x^2 + 7$ (simplifies to 0 or F)
- wa $c$ (sp $c\, a$) $\equiv \exists\, x : \mathbb{Z} \,.\, x > 2 \wedge$ y$^2 + 7 = x^2 + 7$ (yields y $> 2 \vee$ y $< -2$)



## 2 Generalization to program semantics

### 2.0 Conventions

a. Preliminary remark  A familiar "problem": Given a variable $x$, we have $x \in \mathit{Variable}$ (at metalevel) and, for instance, $x \in \mathbb{Z}$ (in the language). Not resolvable without considerable nomenclature, yet clear from the context. Conclusion: let us exploit it rather than lose time over it (at this stage). Remark: similar (more urgent) problem in calculus: how to "save Leibniz" by formalizing  if $y = x^2$, then $\mathrm{d}\, y = 2 \cdot x \cdot \mathrm{d}\, x$  (partial solution 11 years ago)

b. State space

- Not the denotational semantics view where state $s : \mathit{Variable} \to \mathit{Value}$
- State space $\mathbf{S}$ is Cartesian product determined by variable declarations. Henceforth, $s$ is shorthand for the tuple of all program variables. Auxiliary variables ("ghost" or "rigid" variables) appended if necessary.
- Example: `var x : int, b : bool` yields $\mathbf{S} = \mathbb{Z} \times \mathbb{B}$ and $s =$ x, b.
- In what follows, $v$ is a tuple of variables, type $S_v$, in particular $\mathbf{S} = S_s$.

## 2.1 Expressing Floyd-Hoare semantics in terms of program equations

a. Program equations formalizing the intuitive behaviour of command $c$

- $R\,c\,(\grave{}s, s')$ expressing the state change (suitable $R : C \to \mathbf{S}^2 \to \mathbb{B}$)
- $T\,c\,s$ expressing termination of $c$ started in state $s$ (suitable $T : C \to \mathbf{S} \to \mathbb{B}$)
  In further treatment: only guaranteed, not just possible termination
  E.g., *not* as in the example in Gordon's *Specification and Verification 1*,
  which would amount (with our conventions) to $T\,c\,s \;\equiv\; \exists\, s' : \mathbf{S} \,.\, R\,c\,(s, s')$.

b. Formalizing intuitive Floyd-Hoare semantics for weak correctness

- About $\grave{}s$ and $s'$ we know $a[^{s}_{\grave{}s}$ and $R\,c\,(\grave{}s, s')$, no less, no more.
- Therefore: $a[^{s}_{\grave{}s} \wedge R\,c\,(\grave{}s, s') \;\Rightarrow\; p[^{s}_{s'}$ (implicitly quantified over $\grave{}s$ and $s'$).
- Hence $\boxed{\{a\}\;c\;\{p\} \;\equiv\; \forall\, \grave{}s : \mathbf{S} \,.\, \forall\, s' : \mathbf{S} \,.\, a[^{s}_{\grave{}s} \wedge R\,c\,(\grave{}s, s') \;\Rightarrow\; p[^{s}_{s'}} \qquad (3)$

c. Strong correctness: defining $Term$ by $\boxed{Term\,c\,a \;\equiv\; \forall\, s : \mathbf{S} \,.\, a \Rightarrow T\,c\,s} \qquad (4)$

$$\boxed{\begin{array}{l} [a]\;c\;[p] \;\equiv\; \{a\}\;c\;\{p\} \wedge Term\,c\,a \qquad \text{or, blending in (3):} \\ [a]\;c\;[p] \;\equiv\; \forall\, \grave{}s : \mathbf{S} \,.\, \forall\, s' : \mathbf{S} \,.\, a[^{s}_{\grave{}s} \Rightarrow\; T\,c\,\grave{}s \wedge (R\,c\,(\grave{}s, s') \;\Rightarrow\; p[^{s}_{s'}) \qquad (5) \end{array}}$$



## 2.2 Calculating the weakest antecondition

- Calculation

$$\boxed{\begin{array}{ll} & [a]\;c\;[p] \\ \equiv & \langle \text{Definit. (5)} \rangle \quad \forall\, \grave{}s : \mathbf{S} \,.\, \forall\, s' : \mathbf{S} \,.\, a[^{s}_{\grave{}s} \Rightarrow\; T\,c\,\grave{}s \wedge (R\,c\,(\grave{}s, s') \;\Rightarrow\; p[^{s}_{s'}) \\ \equiv & \langle \text{Rdist } \forall/\Rightarrow \rangle \quad \forall\, \grave{}s : \mathbf{S} \,.\, a[^{s}_{\grave{}s} \Rightarrow \forall\, s' : \mathbf{S} \,.\, T\,c\,\grave{}s \wedge (R\,c\,(\grave{}s, s') \;\Rightarrow\; p[^{s}_{s'}) \\ \equiv & \langle \text{Sdist } \forall/\wedge \rangle \quad \forall\, \grave{}s : \mathbf{S} \,.\, a[^{s}_{\grave{}s} \Rightarrow T\,c\,\grave{}s \wedge \forall\, s' : \mathbf{S} \,.\, R\,c\,(\grave{}s, s') \;\Rightarrow\; p[^{s}_{s'} \\ \equiv & \langle \text{Change vars.} \rangle \; \forall\, s : \mathbf{S} \,.\, a \Rightarrow T\,c\,s \wedge \forall\, s' : \mathbf{S} \,.\, R\,c\,(s, s') \;\Rightarrow\; p[^{s}_{s'} \end{array}}$$

- So we proved $\boxed{[a]\;c\;[p] \;\equiv\; \forall\, s : \mathbf{S} \,.\, a \Rightarrow T\,c\,s \wedge \forall\, s' : \mathbf{S} \,.\, R\,c\,(s, s') \;\Rightarrow\; p[^{s}_{s'}}$
  - Observe, as before, that $T\,c\,s \wedge \forall\, s' : \mathbf{S} \,.\, R\,c\,(s, s') \;\Rightarrow\; p[^{s}_{s'}$ is
    at most as strong as any antecondition $a$ and is itself an antecondition
  - Hence $\boxed{\mathrm{wa}\,c\,p \;\equiv\; T\,c\,s \wedge \forall\, s' : \mathbf{S} \,.\, R\,c\,(s, s') \;\Rightarrow\; p[^{s}_{s'}} \qquad (6)$
- Liberal variant: $\boxed{\mathrm{wla}\,c\,p \;\equiv\; \forall\, s' : \mathbf{S} \,.\, R\,c\,(s, s') \;\Rightarrow\; p[^{s}_{s'}} \qquad (7)$
  (shortcut: obtained by substituting $T\,c\,s \;\equiv\; 1$)

### 2.3 Calculating the strongest postcondition

- Calculation

$$
\begin{array}{ll}
[a]\ c\ [p] & \\
\equiv & \langle \text{Def. (3–5)} \rangle \quad Term\, c\, a \wedge \forall\, `s : \mathbf{S} \,.\, \forall\, s' : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s') \Rightarrow p[^{s}_{s'} \\
\equiv & \langle \text{Swap } \forall \rangle \quad Term\, c\, a \wedge \forall\, s' : \mathbf{S} \,.\, \forall\, `s : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s') \Rightarrow p[^{s}_{s'} \\
\equiv & \langle \text{Ldist } \forall/\Rightarrow \rangle \quad Term\, c\, a \wedge \forall\, s' : \mathbf{S} \,.\, \exists\, (`s : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s')) \Rightarrow p[^{s}_{s'} \\
\equiv & \langle \text{Change var} \rangle \quad Term\, c\, a \wedge \forall\, s : \mathbf{S} \,.\, \exists\, (`s : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s)) \Rightarrow p
\end{array}
$$

- So we proved $[a]\ c\ [p] \ \equiv \ Term\, c\, a \wedge \forall\, s : \mathbf{S} \,.\, \exists\, (`s : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s)) \Rightarrow p$
  - Assuming $Term\, c\, a$, observe, as before, that $\exists\, `s : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s)$ is at least as strong as any postcondition $p$ and is itself a postcondition
  - Hence $\mathsf{sp}\, c\, p \ \equiv \ \exists\, `s : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s)$ provided $Term\, c\, a$ (8)
- Liberal variant: $\mathsf{slp}\, c\, p \ \equiv \ \exists\, `s : \mathbf{S} \,.\, a[^{s}_{`s} \wedge R\, c\, (`s, s)$ (9)



# 3 Application to assignment, sequencing, choice and iteration

### 3.0 Assignment revisited (embedded in the general case)

a. We consider (possibly) multiple assignment Let $c := [\![ v := e ]\!]$

- Here $v$ may be a tuple of variables and $e$ a matching tuple of expressions.
- Convenient in calculations: (W.L.O.G.) $s = v +\!\!+ w$ ($w$ rest of variables); similarly $`s = `v +\!\!+ `w$ and $s' = v' +\!\!+ w'$

b. Formalizing intuitive understanding (note simplest choice of bound variables)

$$
\begin{array}{rcll}
R\, c\, (s, s') & \equiv & s' = s[^{v}_{e} & (10) \\
T\, c\, s & \equiv & 1 &
\end{array}
$$

E.g., $R\, [\![ \mathtt{y,j := y+j,j+1} ]\!]\, ((\mathtt{y,j,k}), (\mathtt{y'}, \mathtt{j'}, \mathtt{k'})) \ \equiv \ \mathtt{y'}, \mathtt{j'}, \mathtt{k'} = \mathtt{y} + \mathtt{j}, \mathtt{j} + 1, \mathtt{k}$

c. Weakest ante- and strongest postconditions From (6) and (8) with (10),

$$
\begin{array}{rcll}
\mathsf{wa}\, [\![ v := e ]\!]\, p & \equiv & p[^{v}_{e} & (11) \\
\mathsf{sp}\, [\![ v := e ]\!]\, a & \equiv & \exists\, `v : S_v \,.\, a[^{v}_{`v} \wedge v = e[^{v}_{`v} & (12)
\end{array}
$$

### 3.1 Sequencing

a. Formalization of intuitive understanding of behaviour

$$\begin{array}{rcl} R\,[\![c';c'']\!]\,(`s,s') & \equiv & \exists\, t:\mathbf{S}\,.\,R\,c'\,(`s,t)\wedge R\,c''\,(t,s') \\ T\,[\![c';c'']\!]\,s & \equiv & T\,c'\,s\wedge\forall\, t:\mathbf{S}\,.\,R\,c'\,(s,t)\Rightarrow T\,c''\,t \end{array} \qquad (13)$$

b. Weakest antecondition (strongest postcondition similar) Let $c := [\![c';c'']\!]$ in

$$\begin{array}{rcll} & \mathsf{wa}\,c\,p & & \\ \equiv & \langle\text{Eqn. wa (6)}\rangle & T\,c\,s\wedge\forall\, s':\mathbf{S}\,.\,R\,c\,(s,s')\Rightarrow p[^s_{s'} & \\ \equiv & \langle\text{Def. } R\ (13)\rangle & T\,c\,s\wedge\forall\, s':\mathbf{S}\,.\,\exists\,(t:\mathbf{S}\,.\,R\,c'\,(s,t)\wedge R\,c''\,(t,s'))\Rightarrow p[^s_{s'} & \\ \equiv & \langle\text{Ldist. } \forall/\Rightarrow\rangle & T\,c\,s\wedge\forall\, s':\mathbf{S}\,.\,\forall\, t:\mathbf{S}\,.\,R\,c'\,(s,t)\wedge R\,c''\,(t,s')\Rightarrow p[^s_{s'} & \\ \equiv & \langle\text{Rearrange}\rangle & T\,c\,s\wedge\forall\, t:\mathbf{S}\,.\,R\,c'\,(s,t)\Rightarrow\forall\, s':\mathbf{S}\,.\,R\,c''\,(t,s')\Rightarrow p[^s_{s'} & \\ \equiv & \langle\text{Blend } T\ (13)\rangle & T\,c'\,s\wedge\forall\, t:\mathbf{S}\,.\,R\,c'\,(s,t)\Rightarrow T\,c''\,t\wedge\forall\, s':\mathbf{S}\,.\,R\,c''\,(t,s')\Rightarrow p[^s_{s'} & \\ \equiv & \langle\text{Eqn. wa (6)}\rangle & T\,c'\,s\wedge\forall\, t:\mathbf{S}\,.\,R\,c'\,(s,t)\Rightarrow(\mathsf{wa}\,c''\,p)[^s_t & \\ \equiv & \langle\text{Eqn. wa (6)}\rangle & \mathsf{wa}\,c'\,(\mathsf{wa}\,c''\,p) & \end{array}$$

Remark: Gordon observes that this could not be obtained by $T\,c\,s\ \equiv\ \exists\, t:\mathbf{S}\,.\,R\,c\,(s,t)$



### 3.2 Choice (nondeterministic; deterministic as particular case)

a. Formalizing intuitive understanding Let $ch := [\![\texttt{if}\ []\, i:I\,.\,b_i\ \texttt{->}\ c_i\ \texttt{fi}]\!]$ in

$$\begin{array}{rcl} R\,ch\,(s,s') & \equiv & \exists\, i:I\,.\,b_i\wedge R\,c_i\,(s,s') \\ T\,ch\,s & \equiv & \forall\, i:I\,.\,b_i\Rightarrow T\,c_i\,s \end{array} \qquad (14)$$

Remark: $I$ is just a (finite) indexing set, say, $0\,..\,n-1$ for $n$ alternatives.

b. Weakest ante-, strongest postcondition Let $ch := [\![\texttt{if}\ []\, i:I\,.\,b_i\ \texttt{->}\ c_i\ \texttt{fi}]\!]$ in

$$\begin{array}{rcll} \mathsf{wa}\,ch\,p & \equiv & \langle\text{Eqn. wa (6)}\rangle & T\,ch\,s\wedge\forall\, s':\mathbf{S}\,.\,R\,ch\,(s,s')\Rightarrow p[^s_{s'} \\ & \equiv & \langle\text{Def. } R\ (14)\rangle & T\,ch\,s\wedge\forall\, s':\mathbf{S}\,.\,\exists\,(i:I\,.\,b_i\wedge R\,c_i\,(s,s'))\Rightarrow p[^s_{s'} \\ & \equiv & \langle\text{Sdist } \forall/\Rightarrow\rangle & T\,ch\,s\wedge\forall\, s':\mathbf{S}\,.\,\forall\, i:I\,.\,b_i\wedge R\,c_i\,(s,s')\Rightarrow p[^s_{s'} \\ & \equiv & \langle\text{Shunt, dist.}\rangle & T\,ch\,s\wedge\forall\, i:I\,.\,b_i\Rightarrow\forall\, s':\mathbf{S}\,.\,R\,c_i\,(s,s')\Rightarrow p[^s_{s'} \\ & \equiv & \langle\text{Blend } T\ (14)\rangle & \forall\, i:I\,.\,b_i\Rightarrow T\,c_i\,s\wedge\forall\, s':\mathbf{S}\,.\,R\,c_i\,(s,s')\Rightarrow p[^s_{s'} \\ & \equiv & \langle\text{Eqn. wa (6)}\rangle & \forall\, i:I\,.\,b_i\Rightarrow\mathsf{wa}\,c_i\,p \\ \mathsf{sp}\,ch\,a & \equiv & \langle\text{Similar calc.}\rangle & \exists\, i:I\,.\,\mathsf{sp}\,c_i\,(a\wedge b_i),\ \text{provided}\ \mathit{Term}\,c\,a \end{array}$$

c. Particular case: defining $[\![\texttt{if}\ b\ \texttt{then}\ c'\ \texttt{else}\ c''\ \texttt{fi}]\!] = [\![\texttt{if}\ b\ \texttt{->}\ c'\ \texttt{[]}\ \neg\, b\ \texttt{->}\ c''\ \texttt{fi}]\!]$ yields $\mathsf{wa}\,[\![\texttt{if}\ b\ \texttt{then}\ c'\ \texttt{else}\ c''\ \texttt{fi}]\!]\,p\ \equiv\ (b\Rightarrow\mathsf{wa}\,c'\,p)\wedge(\neg\, b\Rightarrow\mathsf{wa}\,c''\,p)$

### 3.3 Iteration

a. Formalizing intuitive understanding Let $l := [\![\texttt{do}\ b\ \texttt{->}\ c\ \texttt{od}]\!]$ in what follows.

Then $\boxed{l\ =\ [\![\texttt{if}\ \neg b\ \texttt{->}\ \texttt{skip}[]b\ \texttt{->}\ c\,;\ l\ \texttt{fi}]\!]}$ formalizes intuition about behaviour.

b. Calculating $R\,l$, $T\,l$ and $\mathsf{wa}\,l$ Using the earlier results, (head) calculation yields:

$$\begin{aligned} R\,l\,(s,s') &\equiv (\neg b \Rightarrow s = s') \wedge (b \Rightarrow \exists\, t : \mathbf{S}\,.\, R\,c\,(s,t) \wedge R\,l\,(t,s')) && (15)\\ T\,l\,s &\equiv (\neg b \Rightarrow 1) \wedge (b \Rightarrow T\,c\,s \wedge \forall\, t : \mathbf{S}\,.\, R\,c\,(s,t) \Rightarrow T\,l\,t) && (16)\\ \mathsf{wa}\,l\,p &\equiv (\neg b \Rightarrow p) \wedge (b \Rightarrow \mathsf{wa}\,c\,(\mathsf{wa}\,l\,p)) && (17) \end{aligned}$$

Equivalently: $\mathsf{wa}\,l\,p \equiv (\neg b \wedge p) \vee (b \wedge \mathsf{wa}\,c\,(\mathsf{wa}\,l\,p))$. Unfolding suggests defining

$$\begin{aligned} w_{n+1}\,l\,p &\equiv (\neg b \wedge p) \vee (b \wedge \mathsf{wa}\,c\,(w_n\,l\,p))\\ w_0\,l\,p &\equiv \neg b \wedge p \end{aligned}$$

By induction, one can prove $\forall\, n : \mathbb{N}\,.\, w_n\,l\,p \Rightarrow \mathsf{wa}\,l\,p$ so $\boxed{\exists\,(n : \mathbb{N}\,.\, w_n\,l\,p) \Rightarrow \mathsf{wa}\,l\,p}$

c. Bounded nondeterminism: extra requirement $T\,l\,s \Rightarrow \exists\,(n : \mathbb{N}\,.\, d_n\,l\,s)$ where $d_0\,l\,s \equiv \neg b$ and $d_{n+1}\,l\,s \equiv b \Rightarrow T\,c\,s \wedge \forall\, t : \mathbf{S}\,.\, R\,c\,(s,t) \Rightarrow d_n\,l\,t$ ($\#$ steps $\leq n$). Then $\boxed{\mathsf{wa}\,l\,p \equiv \exists\,(n : \mathbb{N}\,.\, w_n\,l\,p)}$ (as in Dijkstra's *A Discipline of Programming*).



# 4 Final remarks and conclusions

### 4.0 Final remarks

- On loops: Exploring $R\,l$, $T\,l$ and $\mathsf{sp}\,l$ is left as an exercise
- General All properties in the literature sampled about wa and sp and their liberal variants can be derived and thereby better understood. Noteworthy: Law of the excluded miracle $\boxed{\neg\,(\mathsf{wa}\,c\,0)}$ is $\boxed{T\,c\,s \Rightarrow \exists\, s' : \mathbf{S}\,.\, R\,c\,(s,s')}$

### 4.1 Conclusions

- Expressing semantics by program equations is the simplest and most direct formalization of intuitive understanding of program behaviour.
  Inspired by analogy with "circuit equations".
  Also: similarities with Hehner's formulation became clear at WG2.1 meeting.
- The separate termination predicate complements the relational part so it can properly deal with nondeterminism and termination.
- All properties of Hoare and Dijkstra semantics are calculated by the formal rules for quantification in the functional predicate calculus.