1983

CSP at WSS

C. A. R. Hoare

University of Wollongong

Recommended Citation
http://ro.uow.edu.au/compsciwp/71

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library:
research-pubs@uow.edu.au
CSP at WSS

C.A.R. Hoare

Programming Research Group
Oxford University
45 Banbury Road
Oxford OX2 6PE

Abstract

This preprint contains a copy of the visuals on Communicating Sequential Processes which were presented at the First Wollongong Summer School on the Science of Programming at Sponar's Chalet, January 31 - February 9, 1983.
COMMUNICATING
SEQUENTIAL
PROCESSES
C.A.R. HOARE
SPONAR'S CHALET
JAN-FEB 1983

1. STOPₐ never does anything.
   a STOPₐ = A

2. If x ∈ VMS
   (x → P) first does x
   and then behaves like P
   a(x → P) = xP

   (coin → STOPₐ VMS) accepts a coin
   before stopping

   (coin → choc → coin → choc → STOP)
   serves two customers

   A symbol denotes a class of event
   in which a process may engage

   coin insertion of coin in vending
   machine VMS

   choc extraction of choc from VMS

   The alphabet of a process is the
   set of events in which we are
   interested, e.g.,

   a VMS = {coin, choc}
   (but not empty coin box)

   RECURSION
   A clock does nothing but tick forever
   a CLOCK = {tick:}

   Let X be the "unknown" behaviour.

   X = (tick → X)

   CLOCK is defined as the only solution

   CLOCK = μ(X: aX = [tick: tick → X])

   compare ROOT = μ(X: X = x² + x - 2)

   WARNING: RHS of equation must be
   guarded; i.e., begin with x ≠ .

   X = X has many solutions

   : μX: X is NOT ALLOWED.
1)  
\begin{align*}
\text{CLOCK} &= \text{(tick \rightarrow \text{CLOCK})} \\
&= \text{(tick \rightarrow (tick \rightarrow \text{CLOCK}))} \\
&= \ldots
\end{align*}

A simple vending machine

\begin{align*}
\text{VMS} &= \text{coin} \rightarrow \text{choc} \rightarrow \text{VMS}
\end{align*}

2)  
\begin{align*}
\text{A change giving machine} \\
\text{CHS} &= \{\text{in1p, out2p, out1p}\} \\
\text{CHS} &= \text{in1p} \rightarrow \text{out2p} \rightarrow \text{out1p} \rightarrow \text{out2p} \rightarrow \text{CHS}
\end{align*}

3)  
\begin{align*}
\text{A more complicated vending machine} \\
\text{VMC} &= \{\text{in2p, in2p, large, small, out1p}\} \\
\text{VMC} &= \text{in2p} \rightarrow (\text{large} \rightarrow \text{VMC}) \\
&\quad \text{small} \rightarrow \text{out1p} \rightarrow \text{VMC} \\
&\quad \text{in1p} \rightarrow (\text{small} \rightarrow \text{VMC}) \\
&\quad \text{in1p} \rightarrow (\text{large} \rightarrow \text{VMC}) \\
&\quad \text{in1p} \rightarrow \text{STOP}
\end{align*}

4)  
\begin{align*}
\text{CHOICE} \\
&\text{Let } \alpha P = \chi Q \\
&\{x, y\} \leq \alpha P \land x \neq y
\end{align*}

Then \((x \rightarrow P \mid y \rightarrow Q)\)

either does \(x\) and then behaves as \(P\)

or does \(y\) and then behaves as \(Q\)

The choice is made by some other process

5)  
\begin{align*}
\text{A machine that offers a choice of goods} \\
\text{VMCT} &= \mu X. \text{coin} \rightarrow (\text{choc} \rightarrow X \upharpoonright \text{buff} \rightarrow X)
\end{align*}

6)  
\begin{align*}
\text{GENERAL CHOICE} \\
&\text{Let } A \text{ be a set of events} \\
&\text{Let } P x \text{ be a process with alphabet } B \\
&\quad \text{(for each } x \text{ in } A) \\
&\quad \text{IF } A \subseteq B \\
&\quad (x : A \rightarrow P x) \text{ first does some } x \text{ in } A \\
&\quad \text{then behaves as } P x
\end{align*}

7)  
\begin{align*}
\text{A process which always does anything desired} \\
\text{\(\text{RUN}_A = A\)} \\
\text{\(\text{RUN}_A = (x : A \rightarrow \text{RUN}_A)\)}
\end{align*}

\text{WARNING: DO NOT INSERT THREE COINS.
All processes (so far) can be expressed by general choice

\[ STOP = (x: \{ \} \rightarrow px) \times (y: \{ \} \rightarrow qy) \]

\[ (b \rightarrow p) = (x: \{ b \} \rightarrow p) \]

\[ (b \rightarrow p | c \rightarrow q) = (x: \{ b, c \} \rightarrow p) \quad if x = b \text{ then } p \text{ else } q \]

\[ \mu x.(x: A \rightarrow p(x,x)) = \]

\[ (x: A \rightarrow p(x, \mu x.(x: A \rightarrow p(x,x)))) \]

**MUTUAL RECURSION**

P and Q may be defined as the unique processes satisfying the simultaneous equations:

\[ P = \text{coin} \rightarrow Q \]

\[ Q = \text{choc} \rightarrow P \]

**Theorem.** \( P = \text{VMS} \)

**Proof.** \( P = \text{coin} \rightarrow \text{choc} \rightarrow P \)

VMS is the only solution of this equation.
\[ (x : A \rightarrow P \Rightarrow) = (y : B \rightarrow Q \Rightarrow) \]
\[ \equiv A = B \land \forall a : A \cdot P a = Q a \]

Corollaries

\[ \text{STOP} \equiv (d \rightarrow p) \]
\[ (c \rightarrow P) \neq (d \rightarrow Q) \text{ if } c \neq d \]
\[ (c \rightarrow P | d \rightarrow Q) = (d \rightarrow Q | c \rightarrow P) \]
\[ (c \rightarrow P) \cdot (c \rightarrow Q) = P = Q \]

\section*{Implementation}

\[ (x : A \rightarrow P \Rightarrow) \text{ is implemented by} \]
\[ \lambda x. \text{if remember } (x, a) \text{ then } P(a) \text{ else BLEEP} \]

\[ \text{STOP} : \lambda x. "\text{BLEEP} \]
\[ (b \rightarrow P) : \text{prefix } (b, P) \]
\[ \text{prefix(s, P)} : \lambda x. \text{if } x = c \text{ then } P \text{ else BLEEP} \]

\[ \text{VMS} := \text{LABEL } . \text{ prefix("coin", prefix("chox", X))} \]

\[ (b \rightarrow P | c \rightarrow Q) \approx \text{choice2 } (b, P, c, Q) \]

\[ \text{ROCKET} := \text{LABEL } . \text{ CT. } \lambda n \]
\[ \begin{cases} 
\text{if } n = 0 \text{ then choice2 } (\text{"around", CT(0)}, \\
\quad \text{"up", CT(1)}) \\
\text{else } \text{choice2 } (\text{"down", CT(n-1)}, \\
\quad \text{"up", CT(n+1)}) \end{cases} \]

\section*{TRACES}

A trace of a process is a sequence of events recording its behaviour up to some moment.

\[ \langle \text{coin, chox, coins} \rangle \text{ is a trace of VMS} \]
\[ \langle \text{coin} \rangle \text{ is also a trace of VMS} \]
\[ \langle \text{coin, chox, coins} \rangle \text{ is a trace of every process } \]
\[ \langle \text{coin, chox, coins, buffer} \rangle \]

are traces of VMCT
\[ \langle \text{ms1, ms1, ms1} \rangle \text{ is a trace of VMC} \]

There is no } x \text{ such that }
\[ \langle \text{ms1, ms1, ms1, ms1} \rangle \text{ is a trace of VMC} \]
CATENATION

s * t is the result of writing t after s.
ex: s = 112233, t = 2233, s * t = 1122332233
s > t = s / t = s  

POWER

t a = t  
t b = t  
(s * t) a = s a t a

Exercise: prove this last law from the preceding ones.

Proofs.

To prove s * t, t s
(1) prove P(s)
(2) prove A(x :: P(x))
(3) prove A(s, t :: P(s, t))

Example
To prove A(s, t :: P(s, t))
(a) prove P(s)
(b) prove A(x :: P(x))
(c) prove A(s, t :: P(s, t))

Let P be a function from symbols to symbols. - total
P(s) is result of applying P to each symbol of s.

double* (1, 5, 3, b) = <2, 10, 6, 2>

Laws

P(s) = s
P(s) = P(s)
P(s * t) = P(s) + P(t)
P(s) = P(s)
P(s) = P(s)

Exercise: give a counter-example.

(3) state a property of P which makes the law valid.
(3) prove it from the previous laws.
MISCELLANEOUS

If $s 
eq e$ then
$s'_n$ is the first element
$s'
$ is result of removing first element

$\#s$ is the length of $s$
$a.b$ is the number of occurrences
of $b$ in $s$
$a.b = \#(s, b)$

REVERSAL

$\overline{e}$ is a written backwards.
$\overline{<a,b,c>} = <c,b,a>$

last($s$) = $\overline{e}$

trace($s$) = $\overline{s}$

Exercise.

Give 10 interesting laws
governing reversal.

IMPLEMENTATION

```
< > NIL
< x > cons (x, NIL)
< x s > cons (x, s)
n s t append (s, t)
 s' car (s)
 s' cdr (s)

THM. $\overline{(s \cdot t)} = \overline{t} \cdot \overline{(s \cdot t)}$

let $f$ be a distributive function.

$f(s) = \text{if } s \neq c \text{ then } f(c)$

else $f(s_1) \cdot f(s)$

$f(s) = \text{if } s = \text{NIL } \text{ then } f(c)$

else append ($f(s_1)$, $f(cdr(s))$)
```

EXAMPLE

```
< > = <>
< x > = if x = A then <x> else <>

((<e, x, A>) (g' A)) = s FA

= if A = x then $\overline{e}$ (s' A) else $\overline{e}$ (g' A)

: restrict $(s, A) = \text{if } s = \text{NIL then NIL}$

else if member (car(s), A)

then cons (car(s), restrict (cdr(s), A))

else restrict (cdr(s), A)
```

EXERCISE

1) Implement $f(s)$ as the LISP function

```
(lisp (f, s)), state explicitly all theorems used.
```

2) Implement $\overline{s}$ as reverse $(s)$, (together

with the relevant theorems)
ORDERING

Set means t begins with a copy of s.

\[ \text{Set} = \exists (u :: s^* u = t) \]

\( s \) is a partial order, i.e.:

- \( \rightarrow \leq s \)
- \( s \leq s \) - reflexive
- \( s \leq t \rightarrow t \leq s \rightarrow s \triangleq t \) - antisymmetric
- \( s \leq t \land t \leq u \rightarrow s \leq u \) - transitive

\( \text{Set} = \exists (u, v :: u^* s^* v = t) \)

\( \text{Set} \) is a partial order.

\[ \text{Set} = \text{Set} \land \# t \leq \# s + n \]

\[ \text{Set} = \text{Set} \]

\( \text{Set} = \exists u \text{ } s^* u \land u \leq t \)

\( \text{Set} = \exists s^* \)

4. A drinks dispenser DD

- \( \text{ADD} = \{ \text{setgin, setwhisky, gin, whisky, com} \} \)
- DD sells gin if the most recent setting was setgin, or whisky

5. VM52 behaves like VM5, except that it accepts up to two coins before dispensing up to two choce.

6. Draw pictures of your answers to 3, 4 and 5.

7. VM5

- \( \mu x . \text{coin} \rightarrow \text{chose} \rightarrow x \)
- VM5 = \( \mu x . \text{chose} \rightarrow \text{coin} \rightarrow x \)

Prove that coin

- VM5 = \( \text{chose} \rightarrow \text{VM5} \)
EXERCISES

1. CTRA = \{up, right\}

CTR A
\[ \begin{array}{c}
\text{ } \\
\text{ } \\
\text{X} \\
\text{ } \\
\text{ } \\
\end{array} \]

CTR B
\[ \begin{array}{c}
\text{ } \\
\text{ } \\
\text{X} \\
\text{ } \\
\text{ } \\
\end{array} \]

CTR C
\[ \begin{array}{c}
\text{ } \\
\text{ } \\
\text{X} \\
\text{ } \\
\text{ } \\
\end{array} \]

2. VMD = \{-1p, 2p, choc\}

choc costs 2p. Any combination and order of coins must be accepted.

3. CTRD = \{up, down, left, right\}

moves on same board as CTRA

RECURSION AGAIN.

To find the solution of the equation
\[ P = F(P) \]
where \( F \) is guarded, and use only approved notations.

Define \( F^0(P) = P \)
\[ F^m(P) = F(F^{m-1}(P)) \]

The required answer is
\[ A(n: n \geq 0: F^n(\text{true})) \]

\[ \text{true, continuous} \]

ASSERTIONS

Let's identify a process with a predicate \( P(br) \) which describes all its possible traces \( br \)

\[ \text{STOP} = br = <> \]

\[ (\text{coin} \rightarrow \text{STOP}) = br = <> v br = \text{coin} \]

\[ \text{CLOCK} = br = F[\text{tick}] = br \]

\[ \text{VHS} = \exists n: n \geq 0: br \in \text{coin}, \text{choc}^n \]

In general
\[ (x: A \rightarrow P(x)) = br = <> \]

\[ v(br \in A \land P(b'_r)[br'/br]) \]

\[ \mu X. F(X) \text{ the unique predicate } P \text{ st } P = F(P) \]

\[ \text{STOP} = br = <> \]

\[ (c \rightarrow P) = br = <> \]

\[ v(b_0 \in c \land P[br'/br]) \]

\[ X = \text{coin} \rightarrow \text{choc} \rightarrow X \]

\[ X = br = <> \]

\[ v(br = \text{coin} \land (\text{choc} \rightarrow X)[br'/br]) \]

\[ \mu \text{true} \]}

\[ X = \text{coin} \rightarrow \text{choc} \rightarrow X \]

\[ P_0 = \text{true} \]

\[ P_1 = br \in \text{coin, choc} \land (\text{coin, choc} \land br) \]

\[ P_2 = br \in \text{coin, choc} \land (\text{coin, choc} \land br) \]

\[ P_3 = br \in \text{coin, choc} \land (\text{coin, choc} \land br) \]

\[ P_n = br \in \text{coin, choc} \land (\text{coin, choc} \land br) \]

\[ \mu X. \text{coin} \rightarrow \text{choc} \rightarrow X \]

\[ A(n: n \geq 0: br \in \text{coin, choc}^n \]

\[ v(\text{coin, choc}^n \land br) \]
8) CONTINUITY

A chain of predicates satisfies
\[ [A: n \geq 0: P_{n+1} \Rightarrow P_n] \]

A predicate transformer \( F \) is continuous if
\[ F(A(n: n \geq 0: P_n)) = A(n: n \geq 0: F(P_n)) \]

for all chains.

Lemma. If \( F \) and \( G \) are continuous, so is \( F \circ G \).

Lemma. If \( F \) is continuous, \( F'(\text{true}) \) is a chain.

If \( F \) is continuous

Let \( X = A(n: n \geq 0: F'(\text{true})) \)

Then \( F(x) = X \)

Proof. LHS =
\[ A(n: n \geq 0: F(F'(\text{true}))) \]
\[ = \text{true} \land A(n: n \geq 1: F'(\text{true})) \]
\[ = \text{RHS.} \]

SPECIFICATIONS.

A process is specified by giving all desired properties of its traces.

"It mustn't accept more than 70 coins."
\[ \text{VMC70} = \neg \exists t^\uparrow \bullet (\text{inp}, \text{m2p}) \leq 70 \]

"It mustn't accept three consecutive pennies."
\[ \text{VMCBar3} = \neg (\text{inp} \circ \text{inp} \circ \text{inp}) \in \text{tr} \]

"An improved VMC."
\[ \text{VMClk} = \text{VMC} \]
\[ \text{AVMCBar3} \land \text{VMC70} \]

ALWAYS.

Let \( P \) be a predicate intended to be true of all traces of a process. It must be satisfied by all prefixes of its traces. So we define:

\[ \Box P = A(s: s \leq \text{tr}: P[s/b^2]) \]

Provided that \( P[s/b^2] \)

\[ \Box P \] uniquely defines a process, with exactly those traces which satisfy it.
9) Examples.

"It mustn't lose money"
\[ \text{NOLOSS} = \Box (\text{br.choc} \leq \text{tr.coin}) \]

"It mustn't cheat the client"
\[ \text{FAIR} = \Box (\text{tr.coin} \leq \text{br.choc} + 1) \]

"It must please owner and client"
\[ \text{NOLOSS} \land \text{FAIR} \]

This specifies VMS

A specification of CTn:
\[ \Box (\text{br.down} \leq \text{br.up} + n) \]

\[ \land \text{last(br) around } \Rightarrow \text{br.down} = \text{br.up} + n \]

Prove VMS \[ \square 0 \leq \text{tr.coin} - \text{tr.choc} < 2 \]

Proof. coin \rightarrow \text{choc} \rightarrow VMS

\[ \land \Box 0 \leq \text{tr.choc} - \text{tr.choc} < 2 \]

\[ = VMS \]

LAWS

\[ \square P = P \]

\[ \Box \Box P = \Box P \]

\[ \Box (P \land Q) = (\Box P) \land (\Box Q) \]

\[ \Box P ; P = \Box (P ; P) \]

\[ \text{tr} = \chi \Rightarrow V \]

\[ (\Box P)[br' / br] = \Box (P[br' / br]) \]

if \ P \ is \ healthy

\[ (\Box P)[f(b_i) / b_i] = \Box (P[f(b_i) / b_i]) \]

AFTER

If s is a trace of P

P/s describes the behaviour of P after it has engaged in s

VMC/\text{cin}2p, small = \text{out}1p \rightarrow VMC

VMC/\text{cin}1p^3 = \text{STOP} \]

LAWS

\[ P / s = P \]

\[ P / (s,t) = (P/s)/t \]

\[ (x : A \rightarrow P_u)/y_1 = P_y \] if \ y \in A

\[ \text{assertion} \]

Provided \ P [s / br]

\[ P / s = P [s \ast br / br] \]

\[ (\Box P)/s = (\Box P/s) \]
Let $P$ be specification of a mechanism driven by "menu-select" keyboard. $\alpha P$ contains a symbol for each key except backspace.

We wish to transform $P$ to a spec.

backable ($P$) with alphabet $\alpha P \cup \{\text{backspace}\}$ which specifies a mechanism which behaves as $P$ except that backspace cancels the effect of the most recent uncancelled symbol in $\alpha P$.

$$\text{backable} (P) = \exists P (\text{clean} \left( \text{br} \right) \lor \text{br}))$$

$$\text{clean} (s) \equiv s \text{ if } s \notin \{\text{backspace}\}$$

$$\text{clean} (s^{<}e^{>}) = \text{clean} ((e^{<}s^{>})) \text{ if } e \in \alpha P$$

$$\text{clean} (s^{<}e^{>}\text{backspace}^{>}t) = \text{clean} (s^{<}t) \text{ for } e \in \alpha P$$

**EXERCISES**

Specify the following predicate transformers:

1. $\text{restart} \in \alpha P$

   restartable ($P$) behaves like $P$; but when restart is pressed, it starts from the beginning again.

2. $\alpha P \cap \{\text{saver, restore}\} = \{\}$

   saveable ($P$) behaves like $P$; but when restore is pressed it returns to the state just before the most recent save.

**IMPLEMENTATION**

back ($Q, P$) behaves like backable ($P$) except that an extra backspace will cause it to behave like $Q$ instead.

back ($Q, P$) = $\lambda x$.

if $x = \text{backspace}$ then $Q$

else if $P(x) = \text{"BLEEP"}$ then $\text{"BLEEP"}$

else back (back ($Q, P$), $P(x)$)

backable ($P$) = $\lambda x$.

if $P(x) = \text{"BLEEP"}$ then $\text{"BLEEP"}$

else back (backable ($P$), $P(x)$)

**INTERSECTION**

Let $\alpha P = \alpha Q = \alpha (P \land Q)$

$P \land Q$ behaves like both $P$ and $Q$.

Each event that occurs requires simultaneous participation of both.

$\text{GRCUST} = (\text{choc} \rightarrow \text{GRCUST})$

$\text{GRCUST} = (\text{coin} \rightarrow \text{choc} \rightarrow \text{GRCUST})$

$\text{GRCUST} \lor \text{VMS} = \mu X. \text{coin} \rightarrow \text{choc} \rightarrow X$

$\text{FOOLC} = (\text{in}t2p \rightarrow \text{large} \rightarrow \text{FOOLC})$

$\text{FOOLC} = (\text{in}1p \rightarrow \text{large} \rightarrow \text{FOOLC})$

$\text{FOOLC} \lor \text{VMC} = \mu X. (\text{in}2p \rightarrow \text{large} \rightarrow X)$

$\text{in}1p \rightarrow \text{STOP})$
CONCURRENCY

\( P \parallel Q \)

\[ \alpha(P\parallel Q) = \alpha P \cup \alpha Q \]

\( \forall x : \alpha P \cup \alpha Q, (P\parallel Q) \) does \( x \) whenever \( P \) does

\( \forall y : \alpha Q - \alpha P, (P\parallel Q) \) does \( y \) whenever \( Q \) does

\( \forall x : \alpha P \cup \alpha Q, (P\parallel Q) \) does \( x \)

whenever both \( P \) and \( Q \) do \( x \)

\[ \text{NOISYVM} = \{ \text{coin, clink, choc, toffee} \} \]

\[ \text{NOISYVM} = \{ \text{coin} \rightarrow \text{clink} \rightarrow \text{choc} \rightarrow \text{NOISYVM} \} \]

\[ \alpha \text{CUST} = \{ \text{coin, startcourse, endcourse, choc, toffee} \} \]

\[ \text{CUST} = \text{coin} \rightarrow (\text{toffee} \rightarrow \text{CUST} \]

\[ \text{startcourse} \rightarrow \text{endcourse} \rightarrow \text{choc} \rightarrow \text{CUST} \]

LAWS.

\( (P\parallel Q)/s = (P/s\#P) \parallel (Q/s\#Q) \)

\( \parallel \) is symmetric and associative.

Let \( P = \lambda x : A \rightarrow P x \)

\( Q = \lambda y : B \rightarrow Q y \)

Then \( P\parallel Q = \lambda z : (A\parallel B \rightarrow z B) \rightarrow \)

\[ \lambda z \in (A\parallel B \rightarrow \lambda z \text{ s.t. } \alpha P \text{ does } \alpha z \land \alpha Q \text{ does } \alpha z) \]

where \( A\parallel B = (X \times R \times B) \times (X \times R \times B) \)

ASSERTION

\( P\parallel Q = \exists k \in ((P\cup Q)/k) \)\]

\( \lambda P \left[ \left( \exists k \in (P\cup Q)/k \right) \right] \)

\( \lambda Q \left[ \left( \exists k \in (P\cup Q)/k \right) \right] \)

EXAMPLE

\[ \alpha P = \{ a, c \} \]

\[ \alpha Q = \{ b, c \} \]

1. \( P\parallel Q = \lambda a \rightarrow (\lambda b \rightarrow c \rightarrow \lambda q\pora\ola Q) \)

2. \( P\parallel b \rightarrow Q = \lambda a \rightarrow (\lambda a \rightarrow c \rightarrow \lambda q\pora\ola Q) \)

3. \( \alpha X \left( \lambda a \rightarrow b \rightarrow c \rightarrow X \right) \)

\( \alpha b \rightarrow a \rightarrow c \rightarrow X \)

\( \lambda (P\parallel Q)/a, b, c \)
Let $n \leq m \& m \not\leq n + k$

$P = \square \text{tr.c} = 2 \text{tr.a} \square \text{tr.a} \& \text{tr.c} < 2$

$Q = \square \text{tr.b} = \text{tr.c}$

$\alpha \left(\text{P} \& \text{Q}\right) = \alpha \text{P} \cup \alpha \text{Q} \cup \{\text{switch}\}$

$(\sim Q)$ first behaves like P.

After switch it behaves like Q.

On each subsequent switch it suspends execution of the current one of \{P, Q\}, and resumes execution of the other at the point it was most recently suspended.

$\vdash \left(\text{P} \& \text{Q}\right)/\langle a, c \rangle =$

$= \square \langle a, c, a \& b \rangle \text{tr.b} \langle a, c, a \& b \rangle < \langle a, c, a \& b \rangle, a$

$= \square \text{tr.b} \leq \text{tr.c} + 1 \leq \text{tr.a} + 1$

**PICTURES.**

$\alpha \text{P} = \{a, b, c\}$

$\alpha \text{Q} = \{b, c, d\}$

$\alpha \left(\text{P} \& \text{Q}\right) = \{a, b, c, d\}$

**THE DINING PHILOSOPHERS.**


\[ P \circ \text{MOD}_{5} \]

\[ \phi \text{PHIL}_i = \{ \text{i sits down, i gets up,} \]
\[ \text{i picks up i, i picks up } \theta 1, \]
\[ \text{i puts down i, i puts down } \theta 1 \} \]

\[ \phi \text{FORK}_i = \{ \text{i picks up i, } \theta 1 \text{ picks up i,} \]
\[ \text{i puts down i, } \theta 1 \text{ puts down i} \} \]

\[ \text{DEADLOCK} \]

\[ \text{Let sad} = \langle 0 \text{ sits down, 1 sits down, ..., 4 sits down,} \]
\[ \theta \text{ picks up } 0, \text{ i picks up } 1, ... \theta \text{ picks up } 4 \rangle \]

\[ \text{COLLEGE } \cup \text{ sad } = \text{ STOP.} \]

\[ \text{SOLUTION} \]

introduce a FOOTMAN

\[ \phi \text{FOOTMAN} = \bigcup_{i=0}^{4} \{ \text{i sits down, i gets up} \} \]

\[ \text{FOOTMAN} = \bigcup_{i=0}^{4} \text{ ups } \frac{i}{2} \text{ downs} \]

where \[ \text{ups} = \frac{i}{2} \text{ br. i gets up} \]
\[ \text{downs} = \frac{i}{2} \text{ br. i sits down} \]

\[ \text{NEW COLLEGE} = \text{COLLEGE} \cup \text{FOOTMAN} \]

\[ \phi \text{PHIL}_i = \mu X. \rho \text{ sits down } \to \text{i picks up i } \to \]
\[ \to \text{i picks up } \theta 1 \to \text{i puts down i } \to \]
\[ \to \text{i puts down } \theta 1 \to \text{i gets up } \to \]
\[ \text{X.} \]

\[ \phi \text{FORK}_i = \mu X. (\text{i picks up i } \to \text{i puts down i } \to \text{X} \]
\[ \text{i puts up } \theta 1 \to \text{i puts down i } \to \text{X} \}

\[ \phi \text{PHILO} = \phi \text{PHIL}_1 \| \phi \text{PHIL}_2 \| ... \| \phi \text{PHIL}_4 \]

\[ \phi \text{FORKS} = \phi \text{FORK}_1 \| \phi \text{FORK}_2 \| ... \| \phi \text{FORK}_4 \]

\[ \text{COLLEGE} = \phi \text{PHILO} \| \phi \text{FORKS} \]

\[ \text{CHANGE OF SYMBOL} \]

\[ \text{Let } F \text{ map } xP \text{ onto } Q \]
\[ P = F^2(Q) \]

means \( P \) performs \( x \) exactly on

\[ \text{these occasions that } Q \text{ performs } F(x). \]
\[ \phi F^2(Q) = F^2(\alpha Q) \]

where \[ F^2(B) = \{ x | F(x) \in B \} \]

To deal with inflation
\[ F(\text{small}) = \text{ large} \]
\[ F(\text{small}) = \text{ small} \]
\[ F(\text{out}) = \text{ out} \]

\[ \text{NEWVHC} = F^2(VHM) \]

\[ g(i \text{ sits down}) = \text{ sits down } \text{ for } 0 \leq i \leq 5 \]
\[ g(i \text{ gets up}) = \text{ gets up } \]

\[ \text{LKY} = g^{-1}(\mu X. \text{ sits down } \to \text{ gets up } \to X) \]
\( F^3(\text{STOP}_P) = \text{STOP} F^2(\text{STOP}) \)
\( F^3(x \to P) = (y \to F^3([x]) \to F^2(P)) \)
\( F^3(x: A \to P_n) = (x: F^3(a) \to (F^3 P + f)) \)
\( (x: F^3(P/s) = F^3(s)/F^3(s) \)
\( f^3(P/F^3(s)) = F^2(P)/s \)
\( F^3(P \parallel Q) = f^3(P) \parallel F^3(Q) \)
\( F^3(\mu X. F) = \mu X. F^2(F) \)
\( g^2(F^3(P)) = (F g)^3(P) \)

**Laws**

\[ P \land P = P \]
\[ P \land Q = Q \land P \]
\[ P \land (Q \lor R) = (P \land Q) \lor R \]

An operator \( F \) is **distributive** if it distributes through \( \land \), i.e.,
\[ F(P \land Q) = F(P) \land F(Q) \]
\[ G(P, Q \land R) = G(P, Q) \land G(P, R) \]
\[ G(P \land Q, R) = G(P, Q) \land G(P, R) \]

All operators so far are distributive, e.g.
\[ x \mapsto (P \land Q) = (x \mapsto P) \land (x \mapsto Q) \]
\[ (x: A \to (P \land Q x)) = (x: A \to P) \land (x: A \to Q) \]

**Warning:** \( \mu X. \) is **not** distributive.

**Implementations**

A non-deterministic process has more than one implementation, e.g.
\[ \text{or} 1(P, Q) = P \]
\[ \text{or} 2(P, Q) = Q \]

Or even allow environment to choose:
\[ \text{or} 3(P, Q) = \lambda x. \text{if } P(x) = \text{"Bleep" then } Q(x) \]
\[ \text{else if } Q(x) = \text{"Bleep" then } P(x) \]
\[ \text{else or3}(P(x), Q(x)) \]

or1 and or2 are efficient but asymmetric.
The set of implementations is symmetric:
\[ \{ \text{or1}(P, Q), \text{or2}(P, Q), \text{or3}(P, Q) \} \]

For \[ \text{or1}(Q, P), \text{or2}(Q, P), \text{or3}(Q, P) \]
PDQ behaves like P or like Q. The choice is made by the environment on the first step only.

\[(c \rightarrow P) \bot (d \rightarrow Q) \equiv (c \rightarrow P) \bot (d \rightarrow Q) \text{ if } c \neq d\]

\[(c \rightarrow P)(e \rightarrow Q) \equiv c \rightarrow (P \lor Q)\]

**Laws**

- is idempotent, symmetric, associative, and distributive, and has unit STOP \(e\).
- \(P \otimes \text{STOP} = P\)

\[(x : A \rightarrow P \otimes) \Pi (y : B \rightarrow Q \otimes) = x : A \lor B \rightarrow Q \otimes \]

**INTERLEAVING**

Let \(s\) be a sequence of sequences \(s_0 \otimes s_1 \otimes \cdots \otimes s_n \otimes \).

\(\pi P s t u\) takes alternate elements from \(s\) & \(t\):

\[\pi P s t u = \text{if } s = \text{ then } t \text{ else } s, u = \pi P (t, u)\]

\(w\) interleaves \((s, w)\) \(\Xi\)

\[\exists t. s = \pi P (t, w) \land v = \gamma s \land w = \gamma t\]

**Assertion**

\[P \parallel Q = \exists (s, t). \otimes \text{ interleaves } (s, t) \land P [s/br] \land Q [t/br]\]

**Concealment**

\(P \setminus C\) behaves like \(P\) except that events in set \(C\) occur autonomously and invisibly:

\[e(P \setminus C) = eP - C\]

Soundproofing

\[VMS = \text{NOISYVM}\{\text{unk, coffee}\}\]

\[((p. x. a \rightarrow c \rightarrow X)(p. x. c \rightarrow b \rightarrow X)) \setminus \{c\}\]

\[e(a \rightarrow c \rightarrow (p. x.(a \rightarrow b \rightarrow c \rightarrow X)) \setminus \{c\}\]

\[e(a \rightarrow p. x.(a \rightarrow b \rightarrow c \rightarrow X)) \setminus \{c\}\]

\[e(b \rightarrow a \rightarrow X)\]
\( P \{ i \} = P \)
\( (P \backslash B) \setminus C = P \setminus (B \cup C) \)
\( (P \cap Q) \setminus C = (P \setminus C) \cap (Q \setminus C) \)
\( \text{stop} \; \setminus A = \text{stop} \setminus C \)
\( (A \rightarrow P) \setminus (A \rightarrow (P \setminus C)) \)
\( \text{AnC = \{ \}} \)
\( (A \rightarrow P) \setminus (A \rightarrow (P \setminus C)) \)
\( \text{AN} \setminus C = \{ \}
\( (A \rightarrow P) \setminus (A \rightarrow (P \setminus C)) \)
\( \text{X} = \{ x, y, \ldots, z \} \)
\( \prod_{w \in X} P_w = P_1 \cap P_2 \cap \ldots \cap P_n \)

**IMPLEMENTATION**

hide \((P, x)\) implements \(P \setminus \{c\}\)

hide \((P, x)\) = if \(P(x)\) then "BLEEP" end

end "BLEEP" else hide \((P(y), x)\)

else +hide \((P(x), x)\)

end

**NOTE** danger of infinite recursion:
such hiding is not allowed.

**PICTURES**

Just remove hidden symbols from arcs.
Single unlabelled arcs may be suppressed:

\[ \begin{align*}
\text{P} & \\
\text{Q} & \\
\text{R} & \\
\text{STOP} & \\
\end{align*} \]

**WARNING:** is NOT ALLOWED

\[ (\mu X. (c \rightarrow X) \mid d \rightarrow \text{STOP}) \]

**ALPHABET EXTENSION**

Let \(x P \cap B = \{\}\)

\(P \cap B\) behaves like \(P\), except that

\(a(P \cap B) = a \cap P \cup B\)

It never performs any action from \(B\)

\(P \cap B\mid B = P\)

Alphabet extension usually implicit:

\(\text{IF } x P \cap \bar{x} Q = \{\}\)

\(P \parallel Q = P \parallel Q\)

(mean: \(P \parallel Q) \parallel (Q \parallel (x P)) = P \parallel Q\)
PROCES S NAMING

A compound event
\[ m.e \]
records performance of e
by a process named m
\[ m:P \]
does \( m.x \) whenever \( P \)
would do \( x \).
\[ m:P = \text{strip}_m^{-1}(P) \]
where \( \text{strip}_m(m.x) = x \)
A pair of vending machines.
\[ \text{left: VMS} \parallel \text{right: VMS} \]

NOTE: \( \text{VMS} \parallel \text{VMS} = \text{VMS} \)

RECURSION

Another definition of ROCKET
\[ Z = (\text{around } \rightarrow Z) \]
\[ \text{up } \rightarrow [m:Z \parallel \text{LOOP}] \]
\[ \text{LOOP} = (\text{up } \rightarrow m.\text{up } \rightarrow \text{LOOP}) \]
\[ \text{down } \rightarrow (m.\text{down } \rightarrow \text{LOOP}) \]
\[ m.\text{around } \rightarrow Z \times \alpha(m:Z) \]
\[ \alpha(Z) = \{ \text{up, down, around} \} \]
\[ \alpha(\text{LOOP}) = \alpha(Z) \cup \alpha(m:Z) \]
\[ = \{ \text{up, down, around, m.up, m.down, m.around} \} \]

LET \( \alpha(m:P) = \text{SWU} \)
\[ [m:P \parallel U] \]
is a process in which
\( U \)
controls \( m:P \)
by actions in \( \alpha(m:P) \),
and these are concealed from environment
\[ [m:P \parallel U] = [((m:P) \parallel U) \setminus \alpha(m:P)] \]
A PILOT flies a rocket named m
\[ [m:ROCKET \parallel \text{PILOT}] \]
To land the rocket she does:
\[ \mu X. (m.\text{down } \rightarrow X) \]
\[ m.\text{around } \rightarrow \ldots \ldots \]
A pilot with two rockets:
\[ [n:ROCKET || \{ m:ROCKET \parallel \text{PILOT} \}] \]

STACK

\[ \alpha(\text{STACK}) = \{ m.1p, \text{out1p}, m.2p, \text{out2p}, \text{isempty} \} \]
\[ \text{STACK} = \text{EMPTY} \]
\[ \text{EMPTY} \& (\text{isempty } \rightarrow \text{EMPTY}) \]
\[ \mid m.1p \rightarrow \text{[n:EMPTY \parallel \text{LOOP1}] \]
\[ \mid m.2p \rightarrow \text{[n:EMPTY \parallel \text{LOOP2}] \]
\[ \text{LOOP1}(m) = (m.1p \rightarrow \text{m.m.1p } \rightarrow \text{LOOP1}) \]
\[ \mid \text{out1p } \rightarrow \text{CONTRACT} \]
\[ \text{CONTRACT}(m) = (m.\text{out1p } \rightarrow \text{LOOP1}) \]
\[ m.\text{out2p } \rightarrow \text{LOOP2} \]
\[ m.\text{isempty } \rightarrow \text{EMPTY} \]
\[ \text{LOOP1}(n) = m.\text{EMPTY} \]
WARNING

RECURSION UNDER HIDDING
   is not guarded.

Let \( F(X) = x \rightarrow (X \setminus \{a\}) \not\in \{a\} \)
\( a \in X = \{a, b, c\} \)
\( F'(x) = a \rightarrow (a \rightarrow (X \setminus \{a\}) \not\in \{a\}) \not\in \{a\} \)
\( = a \rightarrow (X \setminus \{a\}) \not\in \{a\} \)
\( = a \rightarrow X \setminus \{a\} \not\in \{a\} \)
\( = F(X) \)

: For all \( X \), \( F(X) \) is a solution of
\[
Y = F(Y)
\]

i.e., there are many solutions.

\[
F(X)/\langle a \rangle = X \setminus \{a\} \not\in \{a\}
\]
\( X = Y_0 \rightarrow X \)
\( Y = \mu X. \langle X \rangle \rightarrow X \)

REFUSALS.

Let \( x \neq y \), \( x^P \neq x^Q = \{x, y\} \),
\( P = (x \rightarrow P) \), \( Q = (y \rightarrow Q) \)
\( (P \cap Q) \rightarrow P = (x \rightarrow P \rightarrow y \rightarrow Q) \rightarrow (x \rightarrow P) \)
\( = (x \rightarrow (P \cap Q)) \rightarrow P \)
\( (P \cap Q) \rightarrow P = (P \cap (P \cap Q)) \rightarrow P \)
\( = P \cap (y \rightarrow Q \rightarrow x \rightarrow P) \)
\( = P \cap \text{STOP} \)

\( P \cap Q \neq P \cap Q \)
because \( P \cap Q \) can refuse \( \{x\} \)
on its first step
\( P \cap Q \) can't.

Let \( \tau \) be a new event.
\( \mu x. F(x) = (\mu x. x \rightarrow F(x)) \setminus \{\tau\} \)

The recursion fails just when
the hiding fails.

\[ B \cap \text{refusals}(P) \]
means
\[ P \parallel \langle x : B \rightarrow Q \rangle \]

can deadlock immediately.

\[
\text{refusals}(x : A \rightarrow P) = \{B \mid A \cap B = \{x\} \}
\]

\[
\text{refusals}(P \cap Q) = \text{refusals}(P) \cap \text{refusals}(Q)
\]

\[
\text{refusals}(P \parallel Q) = \{x \mid F(x) \cap \text{refusals}(P) \}
\]

\[
\text{refusals}(F^{-1}(P)) = \{x \mid F(x) \cap \text{refusals}(P) \}
\]

\[
\text{refusals}(P \setminus C) = \bigcup_{x \in C} \{x \mid F(x) \cap \text{refusals}(P) \}
\]
provided this is finite union.
LAWS

\[ X \circ \text{refusals}(P) \Rightarrow X \subseteq \text{ref}(P) \]

\[ \{ \} \circ \text{refusals}(P) \]

\[ (X \cup Y) \circ \text{refusals}(P) \Rightarrow X \circ \text{refusals}(P) \]

\[ X \circ \text{refusals}(P) \land X \subseteq \text{ref}(P) \Rightarrow \]

\[ X \cup \{x\} \circ \text{refusals}(P) \]

\[ \forall <x> \circ \text{traces}(P) \]

\[ (s, B) \text{ is an observation of } P \Rightarrow \]

\[ B \circ \text{refusals}(P/s) \land s \circ \text{traces}(P) \]

\[ (s, B) \leq (t, C) \text{ means } \]

\[ s \leq t \land B \subseteq C \]

OBSERVATIONS

ASSERIONS

Labs: identity a process with a
predicate \( P(b, r, f) \) which describes
all its possible observations. (br, ref)

\[ (\omega : A \rightarrow P) = \{ (br \circ <o, A \subseteq \text{ref}(\subseteq P) \}

\[ \forall <o> \circ (br \circ A \land P(br)[br'/br]) \]

STOP \[ a = br \circ \omega \circ \text{ref} \subseteq A \]

\[ (c \circ P) = \{ (br \circ c \circ \omega, c \circ \text{ref} \subseteq P) \]

\[ c \circ (br = c \circ \text{ref} \subseteq P) \]

\[ P/\omega = P[\omega \circ br/br] \]

\[ (P || Q) = \exists X, Y: \text{ref} = X \circ Y; \]

\[ P[X/\text{ref}, br \cup P/br] \]

\[ Q[Y/\text{ref}, br \cup Q/br] \]

\[ P \circ \text{traces}(Q) \]

\[ P \circ Q = P \lor Q \]

\[ P \circ Q = \{ t \circ \text{traces} \rightarrow (P \circ Q) \lor (P \circ Q) \} \]

\[ P \circ Q = \exists s, t: \text{tr interleave } (s, t); \]

\[ P[s/br] \land Q[t/br] \]

\[ P \cap C = \exists s \subseteq C \circ tr; \]

\[ P[s/br], (c \cup \text{ref} / c) / \text{ref} \]

provided that

\[ \text{At} : [tr] \text{tr } \exists C = t \land P[3 / \text{ref}] \]

is \text{FINITE}
EXAMPLES

VM with buffering of coins perhaps.

Let $\text{BAL} = \text{tr.deposit, tr.withdraw}$

$\text{VM} = \square((\text{BAL} = 0 \land \text{coin} \in \overline{\text{ref}}) \lor (\text{BAL} > 0 \land \text{choc} \in \overline{\text{ref}}))$

$\text{VM}_0 = \text{coin} \rightarrow \text{VM}_1$

$\text{VM}_{\text{in}} = \text{choc} \rightarrow \text{VM}_n$

$n(\text{coin} \rightarrow \text{VM}_{\text{in}} \lor \text{choc} \rightarrow \text{VM}_n)$

$\text{VMS} = \text{VM} \rightarrow \text{VM}

\text{i.e. VMS is a valid implementation of VM}$

EXERCISE

A. Write process CD

B. Write a process CDB such that

$\text{CDB} \rightarrow \text{CD}$

COMMUNICATING PROCESSES

A compound event

c.v

denotes communication of a message with value $v$ on channel $c$

$c.E = \{v \mid c.v \in \overline{a} \}$

gives the set of messages which $P$ can communicate on channel $c$

If $E$ is an expression with value $v$

$(c.E \rightarrow P) = (c.v \rightarrow P)$

outputs $v$ on channel $c$

$(c.E \rightarrow P) = (y: \text{strip}^c(\overline{a}(P)) \rightarrow P(\text{strip}_c(y)))$

inputs a message $x$ on channel $c$

Loosely: $c.E \rightarrow P$ means \overline{a}(P) $\neq [\ ]$

EXAMPLES

$\text{CD}$ is a cash dispenser

$\text{aCD} = \{\text{withdraw, deposit}\}$

$\text{BAL} = \text{tr.deposit, tr.withdraw}$

$\text{CD} = \square(\text{deposit} \in \overline{\text{ref}} \land (\text{BAL} \geq 0 \lor \text{withdraw} \in \overline{\text{ref}}))$

EXERCISE

A. Write process CD

B. Write a process CDB such that

$\text{CDB} \rightarrow \text{CD}$

COPY = $\mu X. \text{left}!y \rightarrow \text{right}!y \rightarrow X$

DOUBLE = $\mu X. \text{left}!y \rightarrow \text{right}!y \rightarrow X$

UNPACK = $P_{\text{c>}} = \text{left}! = (\text{right}!)^\ast$

$P_{\text{c>}} \rightarrow P_{\text{c<}}$

$P_{\text{c<}} = \text{right}!x \rightarrow P_{\text{c>}}$

$P_{\text{c<}} = \text{right}!x \rightarrow P_{\text{c>}}$

PACK = $P_{\text{c>}}$

$P_{\text{c<}} = (\text{left}!)^\ast \rightarrow \text{right}!$

$P_{\text{c<}} = \text{left}?!x \rightarrow P_{\text{c<}}$

$P_{\text{c<}} = \text{right}!x \rightarrow P_{\text{c<}}$

SQUASH = $\mu X. \text{left}?!x \rightarrow$

$\text{if } x = \text{"a" then (right}!!x \rightarrow X)$

$\text{else left}?!y \rightarrow$

$\text{if } y = \text{"a" then (right}!!\text{"a"} \rightarrow X)$

$\text{else (right}!!\text{"a"} \rightarrow \text{right}!y \rightarrow X)$
VAR = left?x → VARx
VARx = (left?y → VARY
       | right!x → VARx)

MERGE =
(left?x → right!x → MERGE
  | left?x → right!x → MERGE)

BUFFER = P.x
P.x = left?x → P.x>
P.x> = (left?y → P.x>.seg>
       | right!x → P.x)

STACK = P.x
P.x = (empty → P.x | left?x → P.x)
P.x> =

PIPES

A pipe is a process with only
the two channels left and right
(P>R) is formed by joining right channel
of P to left channel of Q

left→P.right left→Q.right

Communications on connecting channel hidden
Provided that left(P) = left(Q)
(P>R) = (P[mid/right] || Q[mid/left]) \mid

where P[d] = F^2(P)
F(d,v) = cv
F(b,v) = bv for b ≠ d

«(P>R) = «left(P) u «right(» Q
(P>R)>R = P >> (Q>» R)

TRACES

tr.c = strip«(tr.p.c)
If tr.c < left.3, right.3, left.37
then tr.left = <3, 37>
tr.right = <3

Define tr. = set & # b > s + n

COPY = [] tr.left = tr.right

DOUBLE = [double (tr.left)] & tr.right

UNPACK = [] / tr.left = tr.right
   / (trunc (tr.left)) = tr.right.
   where trunc (x) = x.

PACK = [] tr.left = tr.right
   A VS: s = range tr.right: #s = 125

EXAMPLES

QUADRUPLE = DOUBLE >> DOUBLE
LISTING = UNPACK >> PACK
CONWAY = UNPACK >> SQUASH >> PACK
BUFFERED = UNPACK >> BUFFER >> PACK
BUFFER.3 = COPY >> COPY >> COPY
L(5; 7) = \mathbb{E} u. 5 u \land u T u

\left(\begin{array}{l}
(2) \triangleright (2) = 2 \\
\mathbb{E} \cdot \mathbb{E} = \mathbb{E}^{\mathbb{E}}
\end{array}\right)

\text{where } \mathbb{E}^{\mathbb{E}} = \mathbb{E}^2 I^0

\text{If } S \text{ and } T \text{ are grounded, so is } S; T

\begin{align*}
\text{If } P = \square \text{ br. left } R \text{ br. right} \\
Q = \square \text{ br. left } S \text{ br. right}
\end{align*}

then \( P; Q = \square \text{ br. left } (R; S) \text{ br. right} \)

provided that for all \( s, t \)

\{u \mid s R u \& u S t\} is finite.

\text{BUFFER} \gg \text{BUFFER} = \square \text{ br. left } (\mathbb{E} ; \mathbb{E}) \text{ br. right} = \text{BUFFER}

(COPY \gg COPY) = \square \text{ br. left } \mathbb{E} \text{ br. right}

\text{DOUBLE} \gg \text{DOUBLE} = 
\begin{align*}
\square \text{ br. left } (\text{double}^a ; \mathbb{E} ; \text{double}^a ; \mathbb{E}) \text{ br. right} \\
= (\text{double}^a ; \text{double}^a ; \mathbb{E} ; \mathbb{E}) \\
= (\text{quadruple}^a ; \mathbb{E})
\end{align*}
EXAMPLES

VM1 = (coin → choc → SKIP)
VM3 = VM1; VM1; VM1
VM3 = aVM1
AnBCn = μX. (b → SKIP
                               \[a → (X; (c → SKIP))]\)
accepts n a's followed by a b
followed by n c's, for any n
\[∃m:: b < c^n v < b b < c^n v\]
where \(v\) denotes successful termination

POS = (down → SKIP | up → (POS; POS))
POS terminates when number of down's
first exceeds the number of up's
C₀ = (around → C₀ | up → Cₙ)
C₀ = POS; C₀
C₀ = CT₀

LAWS

; is associative with unit SKIP
distribution
\[(a; A → P n); Q = (a; A → (P n; Q))\]

Theorem  C₀ = CT₀
Proof: C₀ satisfies equations of CT₀
Lemma 0) C₀ = (around → C₀ | up → C₁)
      by def C₀
Lemma 1) C₀ = (up → C₁v2 | down → C₀)
Proof: LHS = POS; C₀
       = (down → SKIP | up → POS; POS; C₀)
      def C₀
      = (down → C₀ | up → POS; POS; C₀)
       def POS
      = (down → C₀ | up → POS; POS; C₀)\(v\)
      assoc
      = (down → C₀ | up → POS; C₀v₁)
      def C₀v₁
      = (down → C₀ | up → C₀v₂)
      def C₀v₁
      = RHS