Chapter 3

Equational Laws

Goals

- Present the equational laws (axioms) and give informal justification for them.
- Apply the laws to prove equivalence between agents: two agents have the same behaviour if we can prove them equal using the laws.

1. Classification of Laws

- Static Combinators: $|$, $\backslash L$, $[f]$
  - Transition rules for these combinators are all of the form

$$\frac{E_1 \stackrel{\alpha_1}{\rightarrow} E_1', \ldots, E_m \stackrel{\alpha_m}{\rightarrow} E_m'}{f(E_1, \ldots, E_n) \stackrel{\alpha}{\rightarrow} f(E_1', \ldots, E_n')}$$

  - The combinator f is present in the expression as the outermost combinator after the action α as well as before.
  - Only those components of an agent may change whose actions have contributed to the action of the agent. So, $E_i' = E_i$ if $E_i \stackrel{\alpha_i}{\rightarrow} E_i'$ is not a premise.
- Dynamic Combinators: Prefix, Summation, and Constants
  - An occurrence of the combinator at the outermost level is present before the action and absent afterwards.

Classification of Equational Laws

- Static Laws—laws for static combinators
- Dynamic Laws—laws for dynamic combinators
- The Expansion Law—the law which refers to both static and dynamic combinators.

2. The Dynamic Laws

Proposition 1: Summation Laws

1. $P + Q = Q + P$ — commutativity
2. $P + (Q + R) = (P + Q) + R$ — associativity
3. $P + P = P$ — idempotency
4. $P + \mathbf{0} = P$ — $\mathbf{0}$ is the zero element of $+$

Justifications

- By the intuition that $+$ represents "choice"
- By transitions (or derivatives)

$$E_1 = E_2 \text{ because } \quad E_1 \xrightarrow{\alpha} E' \text{ iff } E_2 \xrightarrow{\alpha} E'$$

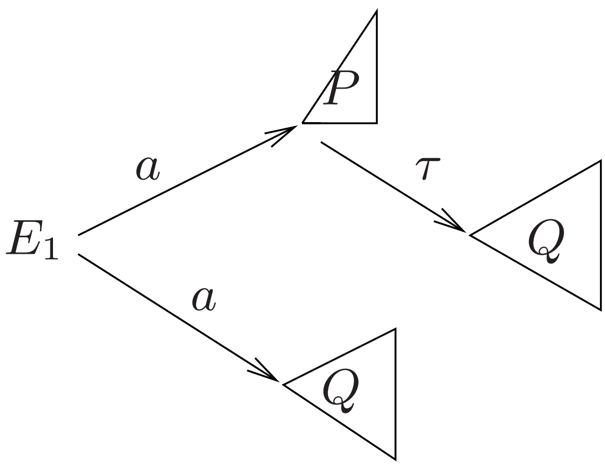
So E_1 and E_2 have the same transition tree.

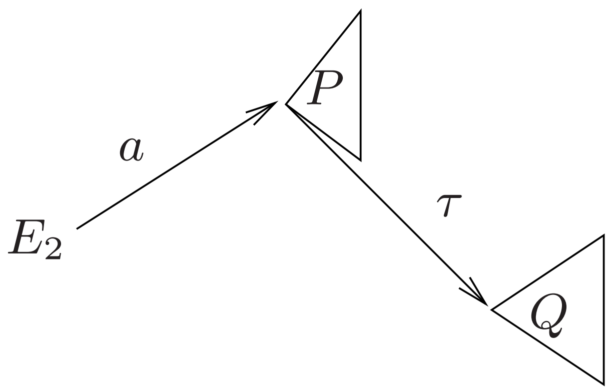
Proposition 2: τ Laws

1. $\alpha.\tau.P = \alpha.P$ — Drop τs except the initial ones
2. $P + \tau.P = \tau.P$
3. $\alpha.(P + \tau.Q) + \alpha.Q = \alpha.(P + \tau.Q)$

Justification

- By the intuition that τ is not observable.
- By derivation trees, let $E1$ and E_2 be the LHS and RHS of (3), respectively





Although $E_1 \xrightarrow{\alpha} Q$, but we do not have $E_2 \xrightarrow{\alpha} Q$.

However, we have $E_2 \xrightarrow{\alpha} P + \tau.Q \xrightarrow{\tau} Q$, i.e. $E_2 \xrightarrow{\alpha}\xrightarrow{\tau} Q$.

Definition: $P \overset{\alpha}{\Rightarrow} P'$ if

$$P(\xrightarrow{\tau})^* \;\xrightarrow{\alpha}\; (\xrightarrow{\tau})^* P'$$

where $(\xrightarrow{\tau})^*$ is the transitive reflexive closure of $\xrightarrow{\tau}$.

Note $\overset{\tau}{\Rightarrow}$ means one or more τ actions.

If we write either (2) or (3) of **Proposition 2** as $E_1 = E_2$, then for any α

$$E_1 \overset{\alpha}{\Rightarrow} E' \text{ iff } E_2 \overset{\alpha}{\Rightarrow} E'$$

But this is too strong to justify (1)

$$\alpha.\tau.P \overset{\alpha}{\Rightarrow} \tau.P \text{ but not } \alpha.P \overset{\alpha}{\Rightarrow} \tau.P$$

Corollary 3: $P + \tau.(P + Q) = \tau.(P + Q)$

Proof:

$$
\begin{array}{lll}
 & P + \tau.(P + Q) & \\
= & P + ((P + Q) + \tau.(P + Q)) & \mathbf{P}.2(2) \\
= & (P + (P + Q)) + \tau.(P + Q) & \mathbf{P}.1(2) \\
= & ((P + P) + Q) + \tau.(P + Q) & \mathbf{P}.1(2) \\
= & (P + Q) + \tau.(P + Q) & \mathbf{P}.1(3) \\
= & \tau.(P + Q) & \mathbf{P}.2(2)
\end{array}
$$

LHS: Actions of P are initially available, and are still available together with those of Q after the τ action.

The corollary says that nothing is gained by making available immediately the capabilities of P if they are available after the initial τ. Or, nothing is lost by deferring the capabilities of P to after the τ action.

Invalid Laws

- $\tau.P = P$.

  If this is valid, then the following is also valid.

  $$a.P + \tau.b.Q = a.P + b.Q$$

  But, we have a problem: LHS may deadlock when a is demanded (this is after the initial τ), but RHS may not deadlock when a is demanded.

- $\tau.P + Q = P + Q$

- $\tau.P + \tau.Q = P + Q$

- $\alpha.(P + Q) = \alpha.P + \alpha.Q$.

  If this is valid, then the following is also valid.

  $$a.(b.P + c.Q) = a.b.Q + a.c.Q$$

  Again, we have a problem: LHS has no deadlock on b and c after action a. But RHS may deadlock if c action is demanded after the initial a.

- $\tau.P + \tau.(P + Q) = \tau.P + \tau.Q$

Recursive Equations

In mathematics we often see equations like

$$x = f(x)$$

v is a solution of this equation iff

$$v = f\{v/x\} \qquad \text{(often written as } v = f(v)\text{)}$$

In general, we have multiple equations like

$$x_i = f_i(x_1, \ldots, x_n) \text{ for } i = 1, \ldots n$$

Again $\tilde{v} = (v_1, \ldots, v_n)$ is a solution of this equation iff

$$v_i = f_i(\tilde{v}/\tilde{x})$$

About solutions of equations

For some equations, there is no solution, e.g.

$$x = x + 1$$

For some equations, there are more than one solutions, e.g.

$$x = x$$

For some equations, there is exactly one solution, e.g.

$$x = 2x$$

Here, $x = 0$

Recursive Agents

We often have recursively defined agents

$$A \stackrel{def}{=} P$$

where P is of the form $E\{A/X\}$. For example, for $E \equiv a.X$, we have

$$A \stackrel{def}{=} a.X\{A/X\}.$$

In this case A is the solution of the equation $X = a.X$

In general

When we write

$$A_i \stackrel{def}{=} E_i\{\tilde{A}/\tilde{X}\}$$

where in E_i at most the variables $\{X_j : j \in I\}$ are free,

we expect $\tilde{A}$ to be a solution of the equation

$$\tilde{X} = \tilde{E}$$

But, when does such family of equations have a unique solution?

$A \stackrel{def}{=} A$ does not define any agent since $X = X$ has any agent as a solution.

Then, under what conditions is there a unique solution to the equation

$$X = E\ ?$$

Conditions for unique solution

Definition:

- X is sequential in E, if it only occurs within Prefix or Summation combinators in E.
- X is guarded in E if each occurrence of X is within some subexpression $l.F$ of E.

Example 1

1. X is sequential but not guarded in
$$\tau.X + a.(b.\mathbf{0}|c.\mathbf{0})$$
2. X is guarded but not sequential in
$$a.X|b.\mathbf{0}$$
3. X is both guarded and sequential in
$$\tau.(P_1 + a.(P_2 + \tau.X))$$

- If X is guarded in E and $A \stackrel{def}{=} E\{A/X\}$, then each recursive call of A must pass through a guard which here is a visible action.
- If X is sequential in E, then each recursive call of A is derived by dynamic transition rules.

Proposition 4

1. If $A \stackrel{def}{=} P$, then $A = P$.
2. Let E_i $(i \in I)$ contain at most the variables $\{X_j : j \in I\}$, and let these variables be guarded and sequential in each E_i. Then,

   $$\text{If } \tilde{P} = \tilde{E}\{\tilde{P}/\tilde{X}\} \text{ and } \tilde{Q} = \tilde{E}\{\tilde{Q}/\tilde{X}\} \text{ then } \tilde{P} = \tilde{Q}$$

   (Part 2 is concerned with the uniqueness of solution of the equations $\tilde{X} = \tilde{E}$, i.e. $X_i = E_i$ for $i \in I$.).

Example 2

$$\begin{array}{ll} A \stackrel{def}{=} a.A' & \qquad B \stackrel{def}{=} c.B' \\ A' \stackrel{def}{=} \overline{c}.A & \qquad B' \stackrel{def}{=} \overline{b}.B \end{array}$$

$$\begin{tikzcd}
 & (A|B')\backslash c \arrow[dl, "\overline{b}"'] \arrow[dr, "a"] & \\
(A|B)\backslash c \arrow[dr, "a"'] & & (A'|B')\backslash c \arrow[dl, "\overline{b}"] \\
 & (A'|B)\backslash c \arrow[uu, "\tau"'] &
\end{tikzcd}$$

From the derivation graphs, we have

1. $(A|B)\backslash c = a.(A'|B)\backslash c = a.\tau.(A|B')\backslash c$.
2. $(A|B')\backslash c$ is the solution of
   $$X = a.\overline{b}.X + \overline{b}.a.X \qquad (\star)$$
   (Formally, we should have said $(A|B')\backslash c$ is the solution of
   $$X = a.\overline{b}.\tau.X + \overline{b}.a.\tau.X$$
   and the τ actions can be removed by 1st τ law (**P**.2(1)).

   $(\star)$ satisfies the condition of **P**.4(2).
3. Let $D \stackrel{def}{=} a.\overline{b}.D + \overline{b}.a.D$.
4. By **P**.4(1), D is a solution of the equation $(\star)$.
5. By **P**.4(2), since $(A|B')\backslash c$ is also a solution of $(\star)$, and since all X are sequential and guarded in $(\star)$, we obtain
   $$(A|B')\backslash c = D$$
6. Thus, $(A|B)\backslash c = a.\tau.(A|B')\backslash c = a.\tau.D = a.D$, where the last step follows from **P**.2(1) (the first τ-law).
7. Hence, $(A|B)\backslash c = a.D$.

3. The Expansion Law

Standard Concurrent Form

We said that a concurrent system is usually composed of several parts such that .... In CCS, this is written as

$$(P_1| \dots, |P_n)\backslash L$$

Example 3. *Jobshop*

$$(Jobber|Jobber|Hammer|Mallet)\backslash L$$

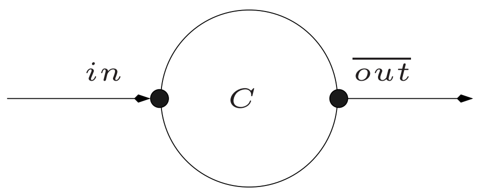
where $L = \{geth, puth, getm, putm\}$

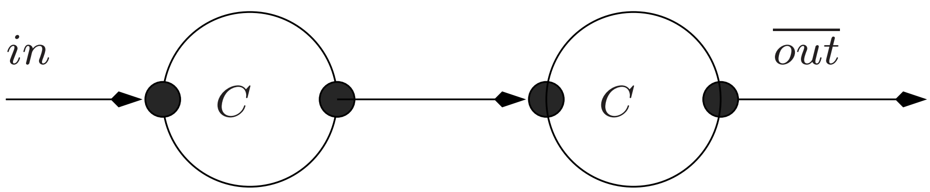
Or

$$(Jobber|Jobber|Sem[geth/get, puth/put]|$$
$$Sem[getm/get, putm/put])\backslash L$$

Example 4. Linked cells

Link two copies of the cell C, with sort $\{in, \overline{out}\}$





$$C \frown C \stackrel{def}{=} (C[mid/out]|C[mid/in])\backslash mid$$

Generally, if we wish to link $\overline{b}$ of P with a of Q, then

$$P \frown Q \stackrel{def}{=} (P[c/b]|Q[c/a])\backslash c \quad \text{where } c \notin \mathcal{L}(P) \cup \mathcal{L}(Q).$$

Standard concurrent form

Definition: A restricted composition of relabelled components is called the Standard Concurrent Form:

$$(P_1[f_1]|\ldots|P_n[f_n])\backslash L$$

Immediate actions of a scf**:**

- If a component, say P_i, has a transition $P_i \xrightarrow{\alpha} P_i'$, and $f_i(\alpha) \notin L \cup \overline{L})$,

$$\begin{array}{c}(P_1[f_1]|\ldots|P_n[f_n])\backslash L \\ \downarrow f_i(\alpha) \\ (P_1[f_1]|\ldots|P_i'[f_i]|\ldots|P_n[f_n])\backslash L\end{array}$$

- If $P_i \xrightarrow{l_1} P_i'$ and $P_j \xrightarrow{l_2} P_j'$ such that $f_i(l_1) = \overline{f_j(l_2)}$,

$$\begin{array}{c}(P_1[f_1]|\ldots|P_n[f_n])\backslash L \\ \downarrow \tau \\ (P_1[f_1]|\ldots|P_i'[f_i]|\ldots|P_j'[f_j]|\ldots|P_n[f_n])\backslash L\end{array}$$

The Expansion Law

Proposition 5: Let $P \equiv (P_1[f_1]| \ldots |P_n[f_n])\backslash L$. Then,

$$\begin{array}{rcl} P & = & \sum\{f_i(\alpha).(P_1[f_1]| \ldots |P_i'[f_i]| \ldots |P_n[f_n])\backslash L : \\ & & \text{for all } P_i \xrightarrow{\alpha} P_i', \text{ where } f_i(\alpha) \notin L \cup \overline{L}\} \\ & + & \sum\{\tau.(P_1[f_1]| \ldots |P_i'[f_i]| \ldots |P_j'[f_j]| \ldots |P_n[f_n])\backslash L : \\ & & \text{for all } P_i \xrightarrow{l_1} P_i' \;\&\; P_j \xrightarrow{l_2} P_j', \text{ where } f_i(l_1) = \overline{f_j(l_2)} \;\&\; i < j\} \end{array}$$

Corollary 6: Let $P \equiv (P_1| \ldots |P_n)\backslash L$. Then,

$$\begin{array}{rcl} P & = & \sum\{\alpha.(P_1| \ldots |P_i'| \ldots |P_n)\backslash L : \\ & & \text{for all } P_i \xrightarrow{\alpha} P_i', \text{ where } \alpha \notin L \cup \overline{L}\} \\ & + & \sum\{\tau.(P_1| \ldots |P_i'| \ldots |P_j'| \ldots |P_n)\backslash L : \\ & & \text{for all } P_i \xrightarrow{l} P_i' \;\&\; P_j \xrightarrow{\overline{l}} P_j', \text{ where } i < j\} \end{array}$$

Remarks

- It is often that high level specification of a system is written in a sequential form, i.e. a summation of prefixed agents, and a lower level description of design is in scf.
- Thus, to show the design is correct w.r.t. the specification, the Expansion Law has to be used again and again (together with other laws, such as τ-laws, constant-law, unique solution-law, etc.).
- When we want to analyse a possible behaviour of a design, the Expansion Law has to be used: this will expand the scf of the design to show all possible interleavings of initial actions and the states (agents) they 'lead' to.

Example 5

Let

$$P_1 \equiv a.P_1' + b.P_1''$$
$$P_2 \equiv \overline{a}.P_2' + c.P_2''$$

What are the initial actions of $P \equiv (P_1|P_2)\backslash a$?

By the Expansion Law:

$$P = b.(P_1''|P_2)\backslash a + c.(P_1|P_2'')\backslash a + \tau.(P_1'|P_2')\backslash a$$

Further let

$$P_3 \equiv \overline{a}.P_3' + \overline{c}.P_3''$$

What are the initial actions of $Q \equiv (P_1|P_2|P_3)\backslash\{a, b\}$?

Let $L = \{a, b\}$, then by the Expansion Law

$$\begin{array}{rcl} Q & = & c.(P_1|P_2''|P_3)\backslash L + \overline{c}.(P_1|P_2|P_3'')\backslash L \\ & + & \tau.(P_1'|P_2'|P_3)\backslash L + \tau.(P_1'|P_2|P_3')\backslash L \\ & + & \tau.(P_1|P_2''|P_3'')\backslash L \end{array}$$

Corollary 7: When $n = 1$

1. $(\alpha.Q)\backslash L = \begin{cases} \mathbf{0} & \text{if } \alpha \in L \cup \overline{L} \\ \alpha.Q\backslash L & \text{otherwise} \end{cases}$
2. $(\alpha.Q)[f] = f(\alpha).Q[f]$
3. $(Q + R)\backslash L = Q\backslash L + R\backslash L$
4. $(Q + R)[f] = Q[f] + R[f]$

Proof of Corollary 7:

Let $P_1 \equiv \alpha.Q$.

(1) Use Corollary 6:

$$
\begin{array}{rl}
P_1\backslash L & = \sum\{\beta.P_1'\backslash L : P_1 \xrightarrow{\beta} P_1', \beta \notin L \cup \overline{L}\} \\
& + \sum_{i\in\emptyset}\{\tau.\cdots\} \\
\text{(by } \mathbf{P}.1(4)) & = \sum\{\beta.P_1'\backslash L : P_1 \xrightarrow{\beta} P_1', \beta \notin L \cup \overline{L}\}) \\
& = \begin{cases} \alpha.Q\backslash L & \text{if } \alpha \notin L \cup \overline{L} \\ \mathbf{0} & \text{otherwise} \end{cases}
\end{array}
$$

(2) Use $\mathbf{P}.5$ and $P\backslash\emptyset = P$

$$
\begin{array}{rl}
P_1[f] & = (P_1[f])\backslash\emptyset \\
& = \sum\{f(\beta).(P_1'[f])\backslash\emptyset : P_1 \xrightarrow{\beta} P_1', f(\beta) \notin \emptyset\} \\
& = f(\alpha).(Q[f])\backslash\emptyset \\
& = f(\alpha).Q[f]
\end{array}
$$

(3) Let $P_1 \equiv Q + R$. Use Corollary 6 and the fact

$Q + R \xrightarrow{\alpha} P'$ iff
$(Q \xrightarrow{\alpha} P')$ or $(R \xrightarrow{\alpha} P')$

$$
\begin{array}{rl}
P_1\backslash L & = \sum\{\alpha.P_1'\backslash L : P_1 \xrightarrow{\alpha} P_1', \alpha \notin L \cup \overline{L}\} \\
& = \sum\{\alpha.P_1'\backslash L : Q \xrightarrow{\alpha} P_1', \alpha \notin L \cup \overline{L}\} \\
& + \sum\{\alpha.P_1'\backslash L : R \xrightarrow{\alpha} P_1', \alpha \notin L \cup \overline{L}\} \\
& = Q\backslash L + R\backslash L
\end{array}
$$

(4) can be proven similarly, but setting $L_1 = \emptyset$.

Example 6:

Consider again $(A|B)\backslash c = a.D$ from Example 2. By the Expansion Law we have

$$\begin{array}{ll} (1) & (A|B)\backslash c = a.(A'|B)\backslash c \\ (2) & (A'|B)\backslash c = \tau.(A|B')\backslash c \\ (3) & (A|B')\backslash c = a.(A'|B')\backslash c + \overline{b}.(A|B)\backslash c \\ (4) & (A'|B')\backslash c = \overline{b}.(A'|B)\backslash c \end{array}$$

So

$$\begin{array}{lll} (A|B)\backslash c & = a.(A'|B)\backslash c & (1) \\ & = a.\tau.(A|B')\backslash c & (2) \\ & = a.(A|B')\backslash c & \mathbf{P}.2(1) \end{array}$$

and

$$\begin{array}{lll} (A|B')\backslash c & = a.\overline{b}.(A'|B)\backslash c + \overline{b}.a.(A'|B)\backslash c & (3),(4),(1) \\ & = a.\overline{b}.\tau.(A|B')\backslash c + \overline{b}.a.\tau.(A|B')\backslash c & (2) \\ & = a.\overline{b}.(A|B')\backslash c + \overline{b}.a.(A|B')\backslash c & \mathbf{P}.2(1) \end{array}$$

Hence, we have $(A|B')\backslash c = a.\overline{b}.(A|B')\backslash c + \overline{b}.a.(A|B')\backslash c$. So, $(A|B')\backslash c$ are D are solutions of

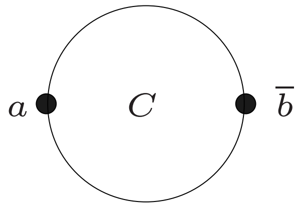
$$X = a.\overline{b}.X + \overline{b}.a.X$$

Since X is sequential and guarded in $a.\overline{b}.X + \overline{b}.a.X$, we have $(A|B')\backslash c = D$ by $\mathbf{P}.4(2)$.

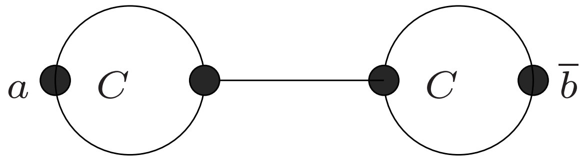
We also have $a.(A|B')\backslash c = a.D$, and hence $a.(A|B)\backslash c = a.D$.

Example 7: Buffers

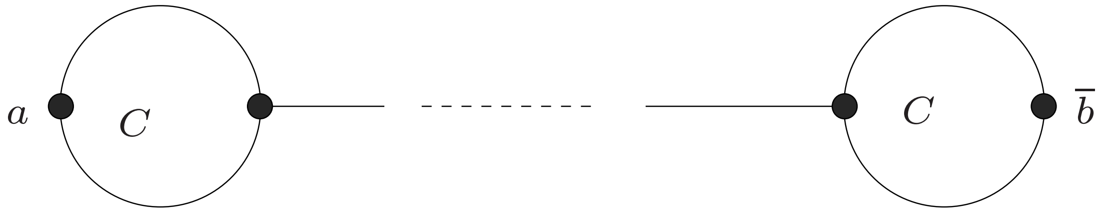
A simplified buffer cell



$$C \overset{def}{=} a.C'$$
$$C' \overset{def}{=} \overline{b}.C$$



$$C \frown C = (C[c/b]|C[c/a])\backslash c$$



$$C^{(1)} \overset{def}{=} C$$
$$C^{(n+1)} \overset{def}{=} C \frown C^{(n)}$$

Theorem: $C^{(n)} = B_n(0)$, where

$$
\begin{array}{lcl}
B_n(0) & \stackrel{def}{=} & a.B_n(1) \\
 & & \\
B_n(k) & \stackrel{def}{=} & a.B_n(k+1) + \\
 & & \overline{b}.B_n(k-1) \qquad (0 < k < n) \\
 & & \\
B_n(n) & \stackrel{def}{=} & \overline{b}.B_n(n-1)
\end{array}
$$

Proof: By induction on n:

For $n = 1$:

$$B_1(0) \stackrel{def}{=} a.B_1(1)$$

$$B_1(1) \stackrel{def}{=} \overline{b}.B_1(0)$$

By the defining equations of C, we have

$$C = B_1(0)$$

Assume $C^{(n)} = B_n(0)$

We prove that $C^{(n+1)} = B_{n+1}(0)$ by induction. We have

$$C^{(n+1)} = C \frown C^{(n)} = C \frown B_n(0)$$

So, if we prove $C \frown B_n(0) = B_{n+1}(0)$ we will be done.

It is enough, by **P.**4, to show that the defining equations of B_{n+1} are satisfied when we replace

$$\begin{array}{lll} B_{n+1}(k) & \text{by } C \frown B_n(k) & (0 \leq k \leq n) \\ B_{n+1}(n+1) & \text{by } C' \frown B_n(n) & \end{array}$$

So the equations to be proved are:

$$\begin{array}{ll} C \frown B_n(0) & = a.(C \frown B_n(1)) \\ C \frown B_n(k) & = a.(C \frown B_n(k+1)) + \overline{b}.(C \frown B_n(k-1)) \\ & \qquad\qquad\qquad\qquad\qquad\qquad (1 \leq k < n) \\ C \frown B_n(n) & = a.(C' \frown B_n(n)) + \overline{b}.(C \frown B_n(n-1)) \\ C' \frown B_n(n) & = \overline{b}.(C \frown B_n(n)) \end{array}$$

First, we use the Expansion Law to compute the left-hand sides:

$$\begin{array}{ll} C \frown B_n(0) & = a.(C' \frown B_n(0)) \\ C \frown B_n(k) & = a.(C' \frown B_n(k)) + \overline{b}.(C \frown B_n(k-1)) \\ & \qquad (1 \leq k \leq n, \text{ for the 2nd, 3rd eq. above}) \\ C' \frown B_n(n) & = \overline{b}.(C' \frown B_n(n-1)) \end{array}$$

Hence, comparing the RHSs of the above sets of equations, we require to show the following

$$\begin{array}{ll} C' \frown B_n(0) & = C \frown B_n(1) \\ C' \frown B_n(k) & = C \frown B_n(k+1) \\ C' \frown B_n(n-1)) & = C \frown B_n(n) \end{array}$$

In fact, the result will follow by $\mathbf{P}.2(1,2)$ if we can prove

$$C' \frown B_n(k) = \tau.(C \frown B_n(k+1)) \quad (0 \leq k \leq n) \quad (*)$$

For this we proceed by induction on k. For $k = 0$, (*) follows by expansion of $C' \frown B_n(0)$. Assume (*) for $k < n - 1$, and consider $k + 1$. By expansion

$$\begin{array}{cl} & C' \frown B_n(k+1) \\ = & \tau.(C \frown B_n(k+2)) + \overline{b}.(C' \frown B_n(k))) \\ = & \tau.(C \frown B_n(k+2)) + \overline{b}.\tau.(C \frown B_n(k+1)) \\ & \text{(induct.)} \\ = & \tau.(C \frown B_n(k+2)) + \overline{b}.(C \frown B_n(k+1)) \\ & \mathbf{P}.2(1) \end{array}$$

Then (*) will follow with k for $k + 1$ if we can absorb the second term in the first.

Such absorption is possible: recall Corollary 3,

$$\tau.(P + Q) + P = \tau.(P + Q)$$

To apply this, we take P to be $\overline{b}.(C \frown B_n(k+1))$ and require that, for some Q,

$$C \frown B_n(k+2) = \overline{b}.(C \frown B_n(k+1)) + Q$$

But (noting that $1 \leq k+2 \leq n$) this is evident from the equations which we obtained by expansion (i.e. those before (*)). We have thus shown

$$C' \frown B_n(k+1) = \tau.(C \frown B_n(k+2))$$

An this completes our inductive proof of (*); it also completes our inductive proof that

$$C^{(n+1)} = B_{n+1}(0)$$

End of the proof.

Example 8. Cyclers

Consider the cycler which fires its ports in a cyclic order:

$$C \stackrel{\mathit{def}}{=} a.C' \qquad C' \stackrel{\mathit{def}}{=} b.C'' \qquad C'' \stackrel{\mathit{def}}{=} c.C$$

We can form a ring $C^{(n)}$ of n cyclers as follows: Assume that $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$ are all distinct; and that addition of integer subscripts is modulo n:

$$C_i \stackrel{\mathit{def}}{=} C[f_i]$$

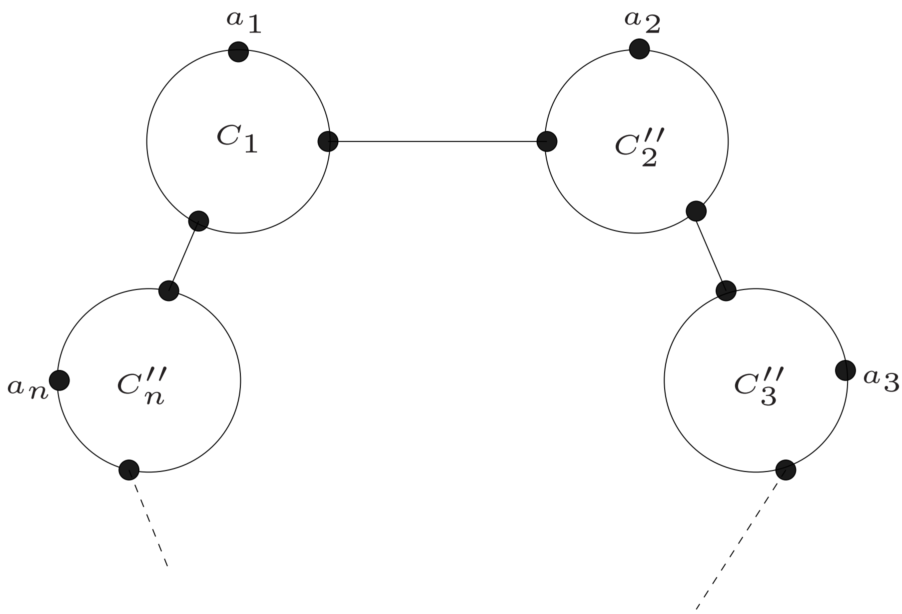
$$C_i' \stackrel{\mathit{def}}{=} C'[f_i]$$

$$C_i'' \stackrel{\mathit{def}}{=} C''[f_i]$$

$$C^{(n)} \stackrel{\mathit{def}}{=} (C_1 | C_2'' | \ldots | C_n'') \backslash L$$

where

$f_i = (a_i/a, b_{i+1}/b, \overline{b_i}/c)$ and $L = \{b_1, \ldots, b_n\}$.



Note that only C_1 is ready to fire its external port; each of the other cyclers is waiting to be enabled by its predecessor.

Use the Expansion Law to show that

$$C^{(n)} = a_1.\cdots.a_n.C^{(n)}$$

where

$$C \stackrel{def}{=} a.C' \qquad C' \stackrel{def}{=} b.C'' \qquad C'' \stackrel{def}{=} c.C$$

$$C_i \stackrel{def}{=} C[f_i]$$

$$C_i' \stackrel{def}{=} C'[f_i]$$

$$C_i'' \stackrel{def}{=} C''[f_i]$$

$$C^{(n)} \stackrel{def}{=} (C_1|C_2''|\ldots|C_n'')\backslash L$$

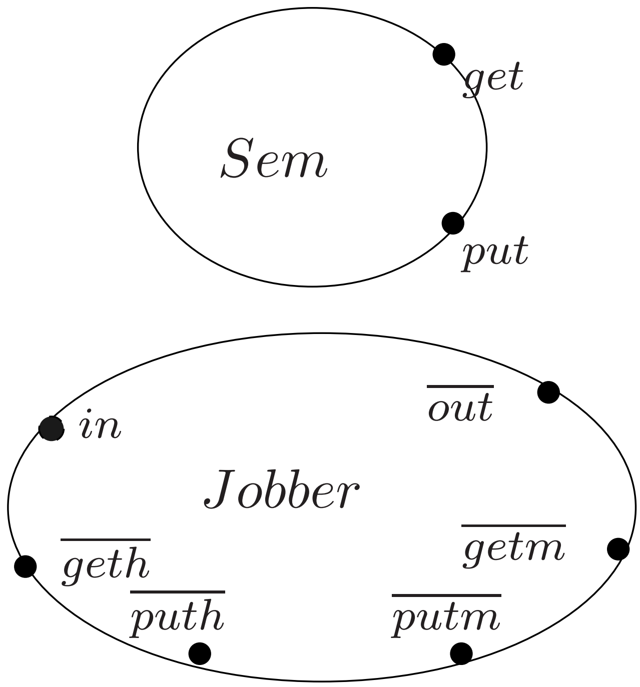
and $\quad f_i = (a_i/a, b_{i+1}/b, \overline{b_i}/c) \quad$ with $\quad L = \{b_1, \ldots, b_n\}$

4. Static laws

- Regard Static Combinators as operations on Flow Graphs.
- Present the static laws
- Justify the laws by intuition
- Justify the laws by flow graphs

Flow Graphs

- A set of Nodes
- Each node has a name and a set of ports with (inner) labels from $\mathcal{L}$.

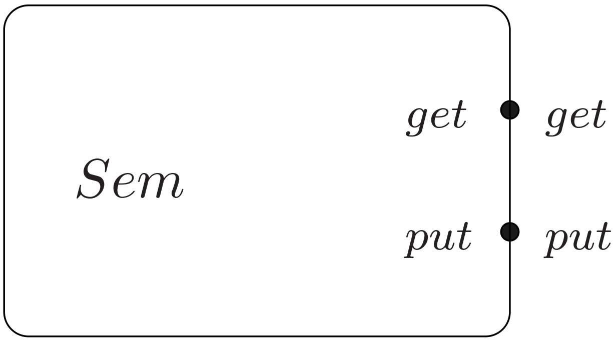


A flow graph is a set of nodes, where pairs of ports are jointed by arcs and some ports are assigned (outer) labels under the following conditions

- If two ports have outer labels l and $\overline{l}$ then they are joined.
- If two ports are joined and one has an outer label l, then the other has outer label $\overline{l}$.

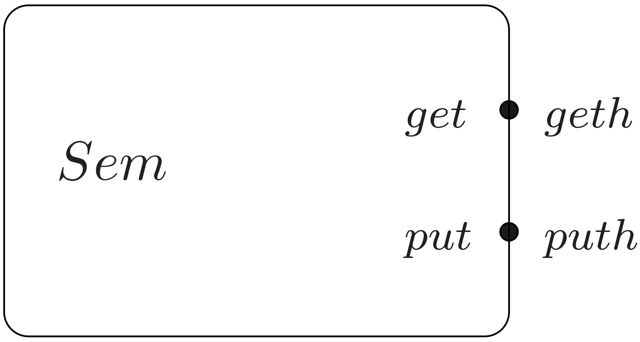
Some examples

- Semaphore



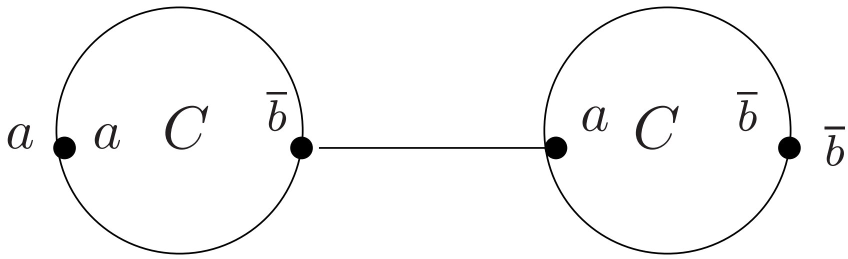
$$Sem \stackrel{def}{=} get.put.Sem$$

- Hammer



$$Hammer \stackrel{def}{=} Sem[geth/get, puth/put]$$

- $C \frown C$



Operations upon flow graphs

- $G|G'$ is formed by joining every pair of ports—one in G and one in G'—which have complementary outer labels.
- $G\backslash L$ is formed by erasing outer labels l and $\bar{l}$ from G, for each $l \in L$.
- $G[f]$ is formed by applying the relabelling function f to all outer labels in G.

Sort $\mathcal{L}(G)$ of G

$$\begin{aligned}
&\mathcal{L}(G|G') = \mathcal{L}(G) \cup \mathcal{L}(G') \\
&\mathcal{L}(G\backslash L) = \mathcal{L}(G) - (L \cup \overline{L}) \\
&\mathcal{L}(G[f]) = f(\mathcal{L}(G))
\end{aligned}$$

The static laws

Proposition 8: Composition laws

1. $P|Q = Q|P$ – commutativity
2. $P|(Q|R) = (P|Q)|R$ – associativity
3. $P|\mathbf{0}$ =P – $\mathbf{0}$ is the unit

Justification

- By the concurrent behaviour of P and Q defined by $|$:

$$LHR \xrightarrow{\alpha} E \text{ iff } RHS \xrightarrow{\alpha} E$$

- By flow graphs.

Proposition 9: Restriction laws

1. $P\backslash L = P$ if $\mathcal{L}(P) \cap (L \cup \overline{L}) = \emptyset$
2. $P\backslash K\backslash L = P\backslash(K \cup L)$
3. $P[f]\backslash L = (P\backslash f^{-1}(L))[f]$
4. $(P|Q)\backslash L = P\backslash L|Q\backslash L$ if
$$\mathcal{L}(P) \cap \overline{\mathcal{L}(Q)} \cap (L \cup \overline{L}) = \emptyset$$

Consider

$$((a.b.\mathbf{0})|(\overline{b}.c.\mathbf{0}))\backslash b \neq (a.b.\mathbf{0})\backslash b|(\overline{b}.c.\mathbf{0})\backslash b$$

Justification

- (1) says that if the labels in L and $\overline{L}$ are not external labels in P, then the Restriction $\backslash L$ is vacuous.
  The special case is $P\backslash\emptyset = P$.
- (2) says that hiding a set of labels followed by hiding another set is the same as hiding the union of the sets.
- (3) says that hiding after relabelling is the same as relabelling after hiding. But, the first is to hide the renamed labels while the second is to hide the old labels.
- (4) says that Hiding L after Composing is the same as Composing after Hiding L in each component, only when the ports labelled by L and $\overline{L}$ are not possible vehicles of communication between P and Q.

Proposition 10: Relabelling laws

1. $P[Id] = P$
2. $P[f] = P[f']$ if $f \restriction \mathcal{L}(P) = f' \restriction \mathcal{L}(P)$
3. $P[f][f'] = P[f' \circ f]$
4. $(P|Q)[f] = P[f]|Q[f]$ if $f \restriction (L \cup \overline{L})$ is one to one, where $L = \mathcal{L}(P|Q)$

Justification

- In (1), Id is the identity function.
- In (2),$f \restriction D$ means the function f restricted to domain D.
- In (3) $(f' \circ f)(l) = f'(f(l))$
- In (4), f is one-one iff $l \neq l'$ implies $f(l) \neq f(l')$. The side condition is needed to ensure that when $[f]$ is applied to P and Q separately, it does not create more complementary port-pairs than existed previously.

  Consider

  $$(a.\mathbf{0}|\overline{b}.\mathbf{0})[c/a, c/b] \neq (a.\mathbf{0})[c/a]|(\overline{b}.\mathbf{0})[c/b]$$

Corollary 11

1. $P[b/a] = P$ if $a, \overline{a} \notin \mathcal{L}(P)$
2. $P\backslash a = P[b/a]\backslash b$ if $b, \overline{b} \notin \mathcal{L}(P)$
3. $P\backslash a[b/c] = P[b/c]\backslash a$ if $b, c \neq a$

Proof:

1. Since $(b/a) \upharpoonright \mathcal{L}(P) = Id \upharpoonright \mathcal{L}(P)$, hence

$$\begin{array}{lll} P[b/a] & = P[Id] & \mathbf{P}.10(2) \\ & = P & \mathbf{P}.10(1) \end{array}$$

2. Let $f = b/a$ and $L = \{b\}$, then $f^{-1}(L) = \{a, b\}$

$$\begin{array}{lll} P\backslash a & = P\backslash a[b/a] & \textbf{Corollary } 11(1) \\ & = P\backslash b\backslash a[b/a] & \mathbf{P}.9(1) \\ & = P\backslash\{a,b\}[b/a] & \mathbf{P}.9(2) \\ & = P[b/a]\backslash b & \mathbf{P}.9(3) \end{array}$$

3. Let $g = b/c$, then $g^{-1}(\{a\}) = \{a\}$

$$\begin{array}{lll} P[g]\backslash a & = P\backslash g^{-1}(\{a\})[g] & \mathbf{P}.9(3) \\ P[b/c]\backslash a & = P\backslash a[b/c] & \end{array}$$

The static laws

Proposition 8: Composition laws

1. $P|Q = Q|P$ – commutativity
2. $P|(Q|R) = (P|Q)|R$ – associativity
3. $P|\mathbf{0}$ =P – $\mathbf{0}$ is an unit

Proposition 9: Restriction laws

1. $P\backslash L = P$ if $\mathcal{L}(P) \cap (L \cup \overline{L}) = \emptyset$
2. $P\backslash K\backslash L = P\backslash(K \cup L)$
3. $P[f]\backslash L = (P\backslash f^{-1}(L))[f]$
4. $(P|Q)\backslash L = P\backslash L|Q\backslash L$ if
$$\mathcal{L}(P) \cap \overline{\mathcal{L}(Q)} \cap (L \cup \overline{L}) = \emptyset$$

Proposition 10: Relabelling laws

1. $P[Id] = P$
2. $P[f] = P[f']$ if $f \restriction \mathcal{L}(P) = f' \restriction \mathcal{L}(P)$
3. $P[f][f'] = P[f' \circ f]$
4. $(P|Q)[f] = P[f]|Q[f]$ if $f \restriction (\mathcal{L}(P|Q) \cup \overline{\mathcal{L}(P|Q)})$ is one to one.

Example 9

The operation of linking is associative:

$$P \frown (Q \frown R) = (P \frown Q) \frown R$$

Proof: Recall

$$P \frown Q \stackrel{def}{=} (P[c/b]|Q[c/a])\backslash c$$

where $c, \overline{c} \notin \mathcal{L}(P) \cup \mathcal{L}(Q)$

First consider $(P \frown Q)[d/b]$, where d is new:

$$\begin{array}{lll} (P \frown Q)[d/b] & = (P[c/b]|Q[c/a])\backslash c[d/b] & \\ & = (P[c/b]|Q[c/a])[d/b]\backslash c & \text{Corollary 11(3)} \\ & = (P[c/b][d/b]|Q[c/a][d/b])\backslash c & \mathbf{P}.10(4) \\ & = (P[c/b]|Q[c/a][d/b])\backslash c & \text{Corollary 11(1)} \\ & = (P[c/b]|Q[c/a, d/b])\backslash c & \mathbf{P}.\ 10(3) \end{array}$$

Hence, since c can be chosen so that $c, \overline{c} \notin \mathcal{L}(R)$,

$$\begin{array}{lll} (P \frown Q) \frown R & = ((P[c/b]|Q[c/a, d/b])\backslash c|R[d/a])\backslash d & \\ & = ((P[c/b]|Q[c/a, d/b])\backslash c|R[d/a]\backslash c)\backslash d & \mathbf{P}.9(1) \\ & = (P[c/b]|Q[c/a, d/b])|R[d/a])\backslash \{c, d\} & \mathbf{P}.9(4) \end{array}$$

Symmetrically, we can reduce $P \frown (Q \frown R)$ to this last expression.

Summary of Chapter 3

- Equational laws—dynamic laws, static laws, and the Expansion Law.
- Justification of the laws by intuition, transitions, transition graphs and by flow graphs, taking into account the character of τ.
- Formal reasoning about the behaviour of agents: two agents have the same observable behaviour if and only if they can be proved equal using the laws of CCS.