Chapter 3. Equational Reasoning

Goals for Chapter 3:

- Present equational laws for CCS with informal justifications.
- Apply the laws to prove equivalence between agents.
- Apply the laws to verify correctness of the design with respect to specification for simple systems.

1 Introduction

In the previous chapters we have introduced the syntax and the operational semantics of CCS. We have also informally discussed the notion of equivalence on agent, i.e. a relation which pairs agents with the same observable behaviour. One way of describing this equivalence is to compare the transition graphs of agents: if two agents have the same transition graphs modulo (a) some τs and (b) change of names at the nodes, then they are equivalent. However, all these definitions are not precise enough.

In this chapter we introduce the elements of the axiomatic (algebraic) theory for CCS, which will tell us precisely when two agents are equivalent. The main part of the theory is the set of equational laws, or simply axioms, for the CCS agents. The axioms represent intuitively the pairs of agent expressions (that themselves represent systems) that have the same observable behaviour. With this we define the equivalence as follows: two agents P and Q are equivalent if and only if P can be re-written into Q, or vice versa, using the equational laws and other laws of equational reasoning.

In the rest of this chapter, the equality symbol $=$ is used as the equality of observable (external) behaviour between agents.

Classification of equational laws

The combinators (operators) of CCS can be divided into

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

$$\frac{E_1 \stackrel{\alpha_1}{\rightarrow} E'_1, \ldots, E_m \stackrel{\alpha_m}{\rightarrow} E'_m}{f(E_1, \ldots, E_n) \stackrel{\alpha}{\rightarrow} f(E'_1, \ldots, E'_n)}$$

  - 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 in the above transition rule.

- Dynamic Combinators: Prefix, Summation, and Constants
  - An occurrence of the combinator at the outermost level is present before the action and absent afterwards.

The equational laws for CCS are classified as follows.

- Dynamic Laws—laws for dynamic combinators.
- Static Laws—laws for static combinators.
- The Expansion Law—the law which concerns 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 $+$

These laws may be justified as follows.

- A choice between several identical options introduces no new possibilities.
- Consider transitions of agents. Let E_1 and E_2 be the LHS and RHS of each of the equations in Proposition 1 respectively. Then we can have

$$E_1 = E_2 \quad \text{because} \quad (E_1 \stackrel{\alpha}{\rightarrow} E' \quad \text{iff} \quad E_2 \stackrel{\alpha}{\rightarrow} E')$$

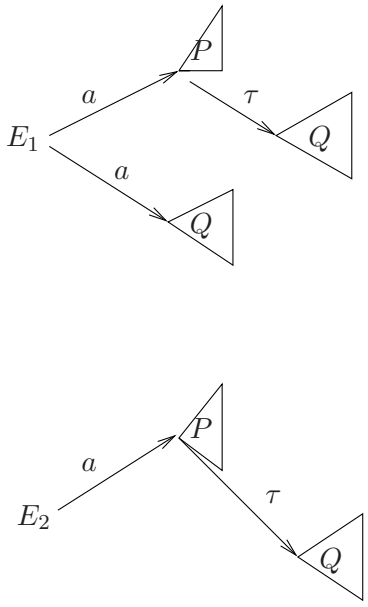
  So, E_1 and E_2 have the "same" transition graphs.

Proposition 2: τ Laws

1. $\alpha.\tau.P = \alpha.P$
2. $P + \tau.P = \tau.P$
3. $\alpha.(P + \tau.Q) + \alpha.Q = \alpha.(P + \tau.Q)$

Below, we try to give an intuitive explanation for the above laws.

The first two τ laws have a natural justification. The first law is explained by the unobservable character of τ. As a result, we can remove all actions τ in a CCS expression as long as they are a part of a subexpression of the form $\alpha.\tau.P$ for any α and any P. The second law simply says that 'adding' P to $\tau.P$ does not offer any more choice of actions of P than $\tau.P$ does. But, the third law is somewhat unclear. We may justify it as follows. Let $E1$ and E_2 be the LHS and RHS of (3) respectively.



We have $E_1 \xrightarrow{\alpha} Q$ but not $E_2 \xrightarrow{\alpha} Q$. However, we have $E_2 \xrightarrow{\alpha} P + \tau.Q \xrightarrow{\tau} Q$. Since τ comes after α and it is unobservable we have a good reason to drop this τ, and thus adopt (3) as a law.

Definition: $P \stackrel{\alpha}{\Rightarrow} P'$ if $P(\xrightarrow{\tau})^* \xrightarrow{\alpha} (\xrightarrow{\tau})^* P'$, where $(\xrightarrow{\tau})^*$ is the transitive reflexive closure of $\xrightarrow{\tau}$.

Note that $\stackrel{\tau}{\Rightarrow}$ means one or more τ transitions.
If we write either (2) or (3) of Proposition 2 as $E_1 = E_2$, then for any α

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

(But this is too strong to justify (1): $\alpha.\tau.P \stackrel{\alpha}{\Rightarrow} \tau.P$ but not $\alpha.P \stackrel{\alpha}{\Rightarrow} \tau.P$.)

Proving agent expressions equal

Now, we demonstrate how to re-write an agent expression into another agent expression using the laws. This amount to showing that the agents are equivalent.

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

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

We can understand Corollary 3 in the following way:

- LHS: Action capabilities of P are initially available, and are still available, together with those of Q, after the τ action.
- RHS: Nothing is lost by deferring the capabilities of P to after the τ action.

Invalid axioms

- $\tau.P = P$. Consider

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

  LHS may deadlock when a is demanded after the LHS does τ, but RHS may not.
- $\tau.P + Q = P + Q$
- $\tau.P + \tau.Q = P + Q$
- $\alpha.(P + Q) = \alpha.P + \alpha.Q$.

  If the above is valid, then so is the following

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

  LHS has no deadlock on actions b and c after action a takes place. But RHS may deadlock if c is demanded after the first a takes place.
- $\tau.P + \tau.(P + Q) = \tau.P + \tau.Q$
- $\tau.P + \tau.Q + \tau.(P + Q) = \tau.P + \tau.Q$
- $\tau.(P + Q) = \tau.(P + Q) + \tau.P + \tau.Q$

Recursive Equations

In mathematics we often see equations like $x = f(x)$. v is a solution of this equation iff $v = f\{v/x\}$. This is often written as $v = f(v)$.

In general, we may have multiple equations like $x_i = f_i(x_1, \ldots, x_n)$, for $i = 1, \ldots, n$. Again $\tilde{v} = (v_1, \ldots, v_n)$ is a solution of this set of equation iff $v_i = f_i(\tilde{v}/\tilde{x})$, for all i.

Solutions of equations

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

   $$x = x + 1$$

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

   $$x = x$$

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

$$x = 2x \text{ has the solution } 0$$

Equations of agent expressions

We often have recursively defined agents $A \stackrel{def}{=} P$, where P is of the form $E\{A/X\}$. For example, $E \equiv a.X$ and $A \stackrel{def}{=} a.A$ (or $A \stackrel{def}{=} E\{A/X\}$). In this case A is a solution of the equation $X = a.X$.

Solving equations of agent expression

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

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

$$X = E \ ?$$

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 $l.F$ of E, where, of course, $l \neq \tau$.

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

$$\tau.(P_1 + a.(P_2 + \tau.X))$$

In general, we have the following.

- 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 is an external action.
- If X is sequential in E, then each recursive call of A is derived by one of the transition rules for dynamic combinators.

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, the following holds.

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

Proposition 4 part 2 is at most times tricky to understand. In very general terms, it says that given a set of agent equations, where all the variables on the RHSs of the equations are guarded and sequential, if there are two (syntactically distinct) sets of agents that are solutions to this set of equations, then the agents from these sets are pairwise equal (according to the axioms and laws of this chapter).

Let us look more carefully at the reading of Proposition 4.2. E_i are agent expressions, for every i in some set of indices I, that may contain at most the agent variables from the set $\{X_j \mid j \in I\}$. For simplicity, let assume that $I = \{0, 1, \ldots, n\}$. So what we have so far is, a set of equations:

$$\begin{aligned} X_0 &= E_0 \\ X_1 &= E_1 \\ &\cdots \\ X_n &= E_n \end{aligned}$$

Such a set of equations is denoted by $\tilde{X} = \tilde{E}$ where n is understood from the context.

Next, the proposition requires that in each E_i all the variables that appear are both guarded and sequential. So, for example, if E_1 contains three variables, say X_0, X_2 and X_1, each of them has to be guarded and sequential. For example, as in $E_1 = a.X_0 + \tau.(b.X_2 + c.X_1)$.

The proposition claims the following:

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

Here, $\tilde{E}\{\tilde{P}/\tilde{X}\}$ denotes our set of expressions E_i with all variables X_i being replaced by agents P_i, all $i \in I$. Recall, an agent is an agent expression that does not involve variables. We assume we have $n + 1$ such agents P_i. Similarly, $\tilde{E}\{\tilde{Q}/\tilde{X}\}$ denotes expressions E_i with all variables X_i being replaced by agents Q_i. We assume we have $n + 1$ such agents Q_i. $\tilde{P} = \tilde{E}\{\tilde{P}/\tilde{X}\}$ means

$$\begin{aligned} P_0 &= E_0\{\tilde{P}/\tilde{X}\} \\ P_1 &= E_1\{\tilde{P}/\tilde{X}\} \\ &\cdots \\ P_n &= E_n\{\tilde{P}/\tilde{X}\} \end{aligned}$$

i.e. all these equations are provable using the axioms and laws from this chapter. Similarly for $\tilde{E}\{\tilde{Q}/\tilde{X}\}$. In particular, for E_1 as above, the second equation directly above reads

$$P_1 = a.P_0 + \tau.(b.P_2 + c.P_1)$$

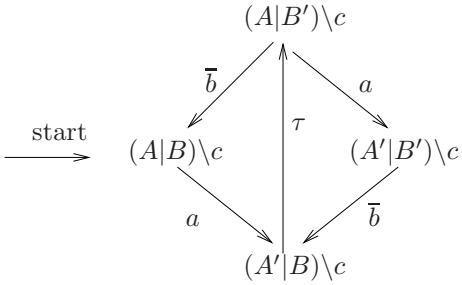
Finally, If $\tilde{P} = \tilde{E}\{\tilde{P}/\tilde{X}\}$ and $\tilde{Q} = \tilde{E}\{\tilde{Q}/\tilde{X}\}$, then the two sets of agents are pairwise equal (i.e. provably equal according to the axioms and laws from this chapter): $\tilde{P} = \tilde{Q}$. The last is an

abbreviation for

$$\begin{aligned} P_0 &= Q_0 \\ P_1 &= Q_1 \\ &\cdots \\ P_n &= Q_n \end{aligned}$$

Example 2 Consider $(A|B)\backslash c$ from the last chapter.

$$A \stackrel{def}{=} a.A' \qquad B \stackrel{def}{=} c.B'$$
$$A' \stackrel{def}{=} \overline{c}.A \qquad B' \stackrel{def}{=} \overline{b}.B$$



From the above derivation graph we have

1. $(A|B)\backslash c = a.(A'|B)\backslash c = a.\tau.(A'|B)\backslash c$ by the Expansion Law below applied twice.
2. $(A|B')\backslash c$ is a solution of $X = a.\overline{b}.\tau.X + \overline{b}.a.\tau.X$ or equivalently, by **P**.2(1), of

   $$X = a.\overline{b}.X + \overline{b}.a.X \qquad (\star)$$

   Note, that X is sequential and guarded in $(\star)$, so the condition of **P**.4(2) is satisfied.
3. We choose D to be a solution for X in $(\star)$. Let $D \stackrel{def}{=} a.\overline{b}.D + \overline{b}.a.D$. By **P**.4(1), D is a solution of the above equation $(\star)$.
4. By **P**.4(2), since also $(A|B')\backslash c$ is also a solution of $(\star)$, and since all X are sequential and guarded in $(\star)$, we obtain

   $$(A|B')\backslash c = D$$
5. With this D, we notice that $a.D$ is the specification of $(A|B)\backslash c$: we can read it out from the derivation graph.
6. Thus, $(A|B)\backslash c = a.(A'|B)\backslash c = a.\tau.(A|B')\backslash c = a.\tau.D = a.D$ by the Expansion Law and the the first τ-law.

Hence, we have proved that $(A|B)\backslash c$, a possible design or an implementation of the system, is correct with respect to the specification $a.D$ of that system.

3 The Expansion Law

A concurrent system is usually composed of several parts such that

- Each part may act independently of (or concurrently with) the other parts,
- The parts may also interact (or communicate) and synchronize with each other.

This means that a concurrent system may have the form $(P_1|\ldots,|P_n)\backslash L$.

Example 3

The *Jobshop* can be represented either as

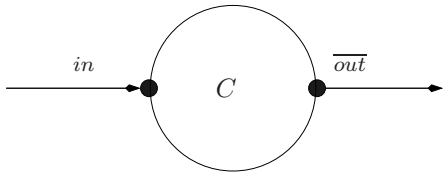
$$(Jobber|Jobber|Hammer|Mallet)\backslash L \quad \text{where } L = \{geth, puth, getm, putm\}$$

or as

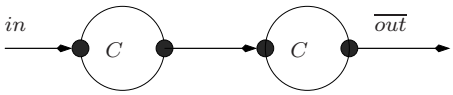
$$(Jobber|Jobber|Sem[geth/get, puth/put]|Sem[getm/get, putm/put])\backslash L$$

Example 4: Linked cells

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



into



The last system is written as (and the linking operator (combinator) $\frown$ is defined by)

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

Generally, if $P \xrightarrow{\overline{b}} P'$ and $Q \xrightarrow{a} Q'$, for some P and Q, then $P \frown Q \stackrel{def}{=} (P[c/b]|Q[c/a])\backslash c$, where c is a fresh name.

Standard concurrent form

Definition: A restricted Composition of relabelled components is called a standard concurrent form (scf):

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

Immediate actions of a scf:

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

$$(P_1[f_1]| \ldots |P_n[f_n])\backslash L \stackrel{f_i(\alpha)}{\longrightarrow} (P_1[f_1]| \ldots |P_i'[f_i]| \ldots |P_n[f_n])\backslash L$$

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

$$(P_1[f_1]| \ldots |P_n[f_n])\backslash L \stackrel{\tau}{\rightarrow} (P_1[f_1]| \ldots |P_i'[f_i]| \ldots |P_j'[f_j]| \ldots |P_n[f_n])\backslash L$$

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

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

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

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

Example 5

Let

$$P_1 \equiv a.P_1' + b.P_1'' \qquad P_2 \equiv \overline{a}.P_2' + c.P_2'' \qquad P \equiv (P_1|P_2)\backslash a$$

Then, by the Expansion Theorem

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

The RHS of the above equation tells us all the initial actions of P.
Further, let

$$P_3 \equiv \overline{a}.P_3' + \overline{c}.P_3'' \qquad Q \equiv (P_1|P_2|P_3)\backslash\{a, b\}$$

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

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

Corollary 7: When $n = 1$ the Expansion Law gives

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.
(1) Hint: Try using Corollary 6 and **P**.1(4).
(2) Use **P**.5 and $P\backslash\emptyset = P$

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

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

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

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

(4) Can be proven similarly, but setting $L_1 = \emptyset$.
Example 6: Consider $(A|B)\backslash c = a.D$ from Example 2. By the Expansion Law we have

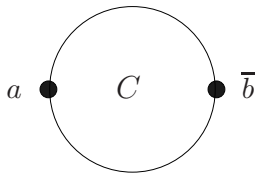
(1) $(A|B)\backslash c = a.(A'|B)\backslash c$
(2) $(A'|B)\backslash c = \tau.(A|B')\backslash c$
(3) $(A|B')\backslash c = a.(A'|B')\backslash c + \overline{b}.(A|B)\backslash c$
(4) $(A'|B')\backslash c = \overline{b}.(A'|B)\backslash c$

So

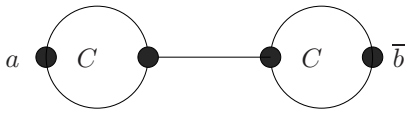
$$
\begin{array}{lll}
(A|B')\backslash c & = a.\overline{b}.(A'|B)\backslash c + \overline{b}.a.(A'|B)\backslash c & (3),(4),(1) \\
& = a.\overline{b}.\tau.(A|B')\backslash c + \overline{b}.a.\tau.(A|B')\backslash c & (2) \\
& = a.\overline{b}.(A|B')\backslash c + \overline{b}.a.(A|B')\backslash c &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

Example 7: Buffers

A simplified buffer cell



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



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

Please, compare the definition of the operator $\frown$ with that on page 58. Note different names of channels.

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

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

$$\begin{aligned}
Buff_n(0) &\stackrel{def}{=} a.Buff_n(1) \\
Buff_n(k) &\stackrel{def}{=} a.Buff_n(k+1) + \overline{b}.Buff_n(k-1) \quad (0 < k < n) \\
Buff_n(n) &\stackrel{def}{=} \overline{b}.Buff_n(n-1)
\end{aligned}$$

Proof: By induction on n:
For $n = 1$:

$$Buff_1(0) \stackrel{def}{=} a.Buff_1(1) \qquad Buff_1(1) \stackrel{def}{=} \overline{b}.Buff_1(0)$$

By the defining equations of C, we have $C = Buff_1(0)$.
Assume $C^{(n)} = Buff_n(0)$. Next, we prove that $C^{(n+1)} = Buff_{n+1}(0)$ as follows.

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

So it will be enough (by **P.**4), if the defining equations of $Buff_{n+1}$ are satisfied when we replace

$$\begin{aligned}
Buff_{n+1}(k) &\quad \text{by } C \frown Buff_n(k) \quad (0 \le k \le n) \\
Buff_{n+1}(n+1) &\quad \text{by } C' \frown Buff_n(n)
\end{aligned}$$

So the equations to be proved are:

$$\begin{aligned}
C \frown Buff_n(0) &= a.(C \frown Buff_n(1)) \\
C \frown Buff_n(k) &= a.(C \frown Buff_n(k+1)) \\
&\quad + \overline{b}.(C \frown Buff_n(k-1)) \quad (1 \le k < n) \\
C \frown Buff_n(n) &= a.(C' \frown Buff_n(n)) \\
&\quad + \overline{b}.(C \frown Buff_n(n-1)) \\
C' \frown Buff_n(n) &= \overline{b}.(C \frown Buff_n(n))
\end{aligned}$$

Use the Expansion Law to compute the left-hand sides:

$$\begin{aligned}
C \frown Buff_n(0) &= a.(C' \frown Buff_n(0)) \\
C \frown Buff_n(k) &= a.(C' \frown Buff_n(k)) \\
&\quad + \overline{b}.(C \frown Buff_n(k-1)) \quad (1 \le k \le n) \\
C' \frown Buff_n(n) &= \overline{b}.(C' \frown Buff_n(n-1))
\end{aligned}$$

Now we can see that the result will follow by **P**.2(1,2), if we can prove

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

For this we proceed by induction on k. For $k = 0$, (*) follows by expansion of $C' \frown Buff_n(0)$. Assume (*) for $k < n_1$, and consider $k+1$. By Expansion Law,

$$\begin{aligned} & C' \frown Buff_n(k+1) \\ = \; & \tau.(C \frown Buff_n(k+2)) + \overline{b}.(C' \frown Buff_n(k))) \\ = \; & \tau.(C \frown Buff_n(k+2)) + \overline{b}.\tau.(C \frown Buff_n(K+1)) \quad \text{(induct.)} \\ = \; & \tau.(C \frown Buff_n(k+2)) + \overline{b}.(C \frown Buff_n(k+1)) \qquad \mathbf{P}.2(1) \end{aligned}$$

Then (*) will follow with k for $k+1$ if we can absorb the second term in the first.
To see that this absorption is possible we recall Corollary 3:

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

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

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

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

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

An this completes our inductive proof of (*); it also completes our inductive proof that $C^{(n+1)} = Buff_{n+1}(0)$.

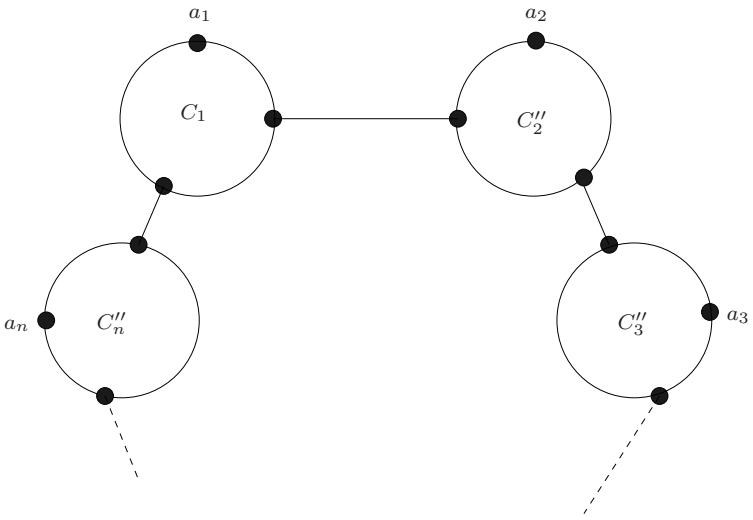
Remark. The above examples show the importance of the Expansion Law. It is often the case that a specification of a system is written in a sequential form, i.e. a summation of prefixed agents, and that a low level implementation is in scf. Thus, to show that the implementation is correct w.r.t. the specification, the Expansion Law has to be used again and again, together with other laws, such as τ-laws, constant-law, unique solution-laws, etc..

Example 8: Cyclers Consider the cycler which fires its ports in a cyclic order:

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

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 (so if $n = 4$ then $n+1$ is 1):

$$\begin{array}{ll} C_i \stackrel{def}{=} C[f_i] & C'_i \stackrel{def}{=} C'[f_i] \\ C''_i \stackrel{def}{=} C''[f_i] & f_i = (a_i/a, b_{i+1}/b, \overline{b_i}/c) \\ C^{(n)} \stackrel{def}{=} (C_1|C''_2|\ldots|C''_n)\backslash L & L = \{b_1, \ldots, b_n\} \end{array}$$



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.

Try to prove, using the Expansion Law and other laws, that

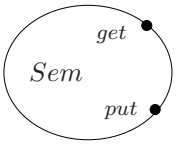
$$C^{(n)} \;=\; a_1.a_2.\cdots.a_n.C^{(n)}$$

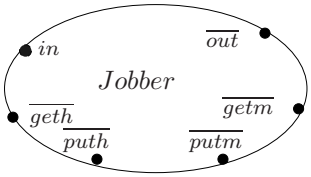
4 Static Laws

- 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 $\mathcal{L}$.





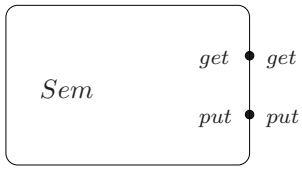
Flow graph

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 $\overline{l}$.

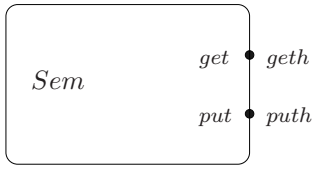
Examples

- Semaphore



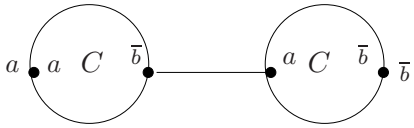
$$Sem \stackrel{def}{=} get.put.Sem$$

- Hammer



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

- $C \frown C$



Operations on 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 $\overline{l}$ 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:

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

4.1 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

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 $\mathcal{L}(P) \cap \overline{\mathcal{L}(Q)} \cap (L \cup \overline{L}) = \emptyset$

Note that $(a.b.\mathbf{0} \mid \overline{b}.c.\mathbf{0})\backslash b \neq (a.b.\mathbf{0})\backslash b \mid (\overline{b}.c.\mathbf{0})\backslash b$. This is because $(a.b.\mathbf{0} \mid \overline{b}.c.\mathbf{0})\backslash b \xrightarrow{a} (b.\mathbf{0} \mid \overline{b}.c.\mathbf{0})\backslash b \xrightarrow{\tau} (\mathbf{0} \mid c.\mathbf{0})\backslash b$ but $(a.b.\mathbf{0})\backslash b \mid (\overline{b}.c.\mathbf{0})\backslash b \xrightarrow{a} (b.\mathbf{0})\backslash b \mid (\overline{b}.c.\mathbf{0})\backslash b \xrightarrow{\tau}\!\!\!\!\!\!\rightarrow$.

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. A 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 two 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 restricting L after composing is the same as composing after restricting L in each component, only when the ports labelled by L and $\overline{L}$ cannot be communicated upon by P and Q.

Proposition 10: Relabelling laws

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

Justification

- In (1), Id is the identity function.
- In (2),$f \upharpoonright D$ 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.

Note that $(a.\mathbf{0} \mid \overline{b}.\mathbf{0})[c/a, c/b] \neq (a.\mathbf{0})[c/a] \mid (\overline{b}.\mathbf{0})[c/b]$. Explain, why this is true.

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 $g^{-1}(\{a\}) = \{a\}$

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

Example 9 Linking is associative:

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

Proof: Recall that $P \frown Q \stackrel{def}{=} (P[c/b]|Q[c/a])\backslash c$, where $c, \overline{c} \notin \mathcal{L}(P) \cup \mathcal{L}(Q)$.

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

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

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

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

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

5 Worked out Examples

Problem: Prove that the following agents are equal. Mention any laws you use.

$$a.(c.\mathbf{0} + \tau.(\tau.b.\mathbf{0} + c.\mathbf{0})) + a.(b.\tau.\mathbf{0} + \tau.(b.\mathbf{0} + c.\mathbf{0}))$$
$$a.(b.\mathbf{0} + c.\tau.\mathbf{0}) + a.b.\mathbf{0} + a.(\tau.b.\mathbf{0} + b.\mathbf{0} + \tau.(\tau.b.\mathbf{0} + c.\mathbf{0}))$$

Answer: $a.(c.\mathbf{0} + \tau.(\tau.b.\mathbf{0} + c.\mathbf{0})) + a.(b.\tau.\mathbf{0} + \tau.(b.\mathbf{0} + c.\mathbf{0}))$ equals

$$\begin{aligned}
&= a.(\tau.b.\mathbf{0} + c.\mathbf{0})) + a.(b.\tau.\mathbf{0} + \tau.(b.\mathbf{0} + c.\mathbf{0})) && \text{by Corollary 3 and 1st } \tau\text{-law} \\
&= a.(\tau.b.\mathbf{0} + c.\mathbf{0})) + a.(b.\mathbf{0} + c.\mathbf{0}) && \text{by Corollary 3 and 1st } \tau\text{-law} \\
&= a.(c.\mathbf{0} + \tau.b.\mathbf{0}) + a.b.\mathbf{0} + a.(b.\mathbf{0} + c.\mathbf{0}) && \text{by 3rd } \tau\text{-law} \\
&= a.(b.\mathbf{0} + c.\mathbf{0}) + a.b.\mathbf{0} + a.(c.\mathbf{0} + \tau.b.\mathbf{0}) && \text{by commutativity of } + \\
&= a.(b.\mathbf{0} + c.\tau.\mathbf{0}) + a.b.\mathbf{0} + a.\tau.(c.\mathbf{0} + \tau.b.\mathbf{0}) && \text{by 1st } \tau\text{-law} \\
&= a.(b.\mathbf{0} + c.\tau.\mathbf{0}) + a.b.\mathbf{0} + a.(\tau.b.\mathbf{0} + \tau.(c.\mathbf{0} + \tau.b.\mathbf{0})) && \text{by Corollary 3} \\
&= a.(b.\mathbf{0} + c.\tau.\mathbf{0}) + a.b.\mathbf{0} + a.(\tau.b.\mathbf{0} + b.\mathbf{0} + \tau.(c.\mathbf{0} + \tau.b.\mathbf{0})) && \text{by 2nd } \tau\text{-law} \\
&= a.(b.\mathbf{0} + c.\tau.\mathbf{0}) + a.b.\mathbf{0} + a.(\tau.b.\mathbf{0} + b.\mathbf{0} + \tau.(\tau.b.\mathbf{0} + c.\mathbf{0})) && \text{by commutativity of } +
\end{aligned}$$

Problem: Let agents P, Q and R be as follows:

$$\begin{aligned}
P &\stackrel{def}{=} a.\overline{d}.b.\overline{f}.\mathbf{0} \\
Q &\stackrel{def}{=} a.\overline{e}.f.c.\mathbf{0} \\
R &\stackrel{def}{=} d.e.\mathbf{0}
\end{aligned}$$

Use the Expansion Law, and other laws of CCS as necessary, to obtain a new agent T such that $T = (P|Q|R) \setminus \{d, e, f\}$ and T uses only $\mathbf{0}$, prefixing with visible actions and sum combinators (and no prefixing with τ). **Note**: To keep the notation under control, $\setminus L$ is always applied to a parallel composition of three agents and it is enclosed in $(\ldots)$.

Answer: Let $L = \{d, e, f\}$. Then $(P|Q|R) \setminus L$ equals to

$$
\begin{aligned}
& a.(\overline{d}.b.\overline{f}.\mathbf{0}|Q|R \setminus L) + a.(P|\overline{e}.f.c.\mathbf{0}|R \setminus L) \\
= \;& a.(\tau.(b.\overline{f}.\mathbf{0}|Q|e.\mathbf{0} \setminus L) + a.(\overline{d}.b.\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|R \setminus L)) + \\
& a.a.(\overline{d}.b.\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|R \setminus L) \\
= \;& a.[\tau.(b.(\overline{f}.\mathbf{0}|Q|e.\mathbf{0} \setminus L) + a.(b.\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|e.\mathbf{0} \setminus L)) + a.\tau.(b.\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|e.\mathbf{0} \setminus L)] + \\
& a.a.\tau.(b.\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|e.\mathbf{0} \setminus L) \\
= \;& a.[\tau.(b.a.(\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|e.\mathbf{0} \setminus L) + a.(b.(\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|e.\mathbf{0} \setminus L) + \tau.(b.\overline{f}.\mathbf{0}|f.c.\mathbf{0}|\mathbf{0} \setminus L))) + \\
& a.(b.\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|e.\mathbf{0} \setminus L)] + a.a.(b.\overline{f}.\mathbf{0}|\overline{e}.f.c.\mathbf{0}|e.\mathbf{0} \setminus L) \\
= \;& \text{Now, all subterms with parallel composition are rewritten into sequential subterms} \\
= \;& a.[\tau.(b.a.(\tau.\tau.c.\mathbf{0} + \tau.\tau.c.\mathbf{0}) + a.(b.(\tau.\tau.c.\mathbf{0} + \tau.\tau.c.\mathbf{0}) + \tau.b.\tau.c.\mathbf{0})) + a.(b.(\tau.\tau.c.\mathbf{0} + \tau.\tau.c.\mathbf{0}) + \\
& \tau.b.\tau.c.\mathbf{0})] + a.a.(b.(\tau.\tau.c.\mathbf{0} + \tau.\tau.c.\mathbf{0}) + \tau.b.\tau.c.\mathbf{0})) \\
= \;& a.[\tau.(b.a.\tau.\tau.c.\mathbf{0} + a.(b.\tau.\tau.c.\mathbf{0} + \tau.b.\tau.c.\mathbf{0})) + a.(b.\tau.\tau.c.\mathbf{0} + \tau.b.\tau.c.\mathbf{0})] + \\
& a.a.(b.\tau.\tau.c.\mathbf{0} + \tau.b.\tau.c.\mathbf{0})) \\
= \;& a.[\tau.(b.a.c.\mathbf{0} + a.(b.c.\mathbf{0} + \tau.b.c.\mathbf{0})) + a(b.c.\mathbf{0} + \tau.b.c.\mathbf{0})] + \\
& a.a.(b.c.\mathbf{0} + \tau.b.c.\mathbf{0})) \\
= \;& a.[\tau.(b.a.c.\mathbf{0} + a.\tau.b.c.\mathbf{0}) + a.\tau.b.c.\mathbf{0}] + a.a.\tau.b.c.\mathbf{0}) \\
= \;& a.[\tau.(b.a.c.\mathbf{0} + a.b.c.\mathbf{0}) + a.b.c.\mathbf{0}] + a.a.b.c.\mathbf{0} \\
= \;& a.[\tau.(b.a.c.\mathbf{0} + a.b.c.\mathbf{0})] + a.a.b.c.\mathbf{0} \;=\; a.(b.a.c.\mathbf{0} + a.b.c.\mathbf{0}) + a.a.b.c.\mathbf{0} \;=\; T
\end{aligned}
$$

Hence $T \stackrel{def}{=} a.(b.a.c.\mathbf{0} + a.b.c.\mathbf{0}) + a.a.b.c.\mathbf{0}$.

6 Summary

- Equational laws—dynamic laws, static laws, the Expansion Law and τ laws.
- Justification of the laws by intuition, transitions, transition graphs and by flow graphs.
- 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.

7 Exercises

1. Use the Expansion Law to prove the following equations:

   (a) $(\alpha.Q)\backslash L = \begin{cases} \mathbf{0} & \text{if } \alpha \in L \cup \overline{L} \\ \alpha.Q\backslash L & \text{otherwise} \end{cases}$

(b) $(\alpha.Q)[f] = f(\alpha).Q[f]$

(c) $(Q + R)\backslash L = Q\backslash L + R\backslash L$

(d) $(Q + R)[f] = Q[f] + R[f]$

2. Prove the $M = Spec$ where

$$\begin{array}{lcl} M & \stackrel{def}{=} & (P_1|P_2|Sem)\backslash\{get, put\} \\ P_1 & \stackrel{def}{=} & \overline{get}.c_1.\overline{put}.P_1 \\ P_2 & \stackrel{def}{=} & \overline{get}.c_2.\overline{put}.P_2 \\ Sem & \stackrel{def}{=} & get.put.Sem \\ Spec & \stackrel{def}{=} & \tau.c_1.Spec + \tau.c_2.Spec \end{array}$$

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

$$C \stackrel{def}{=} a.C' \quad C' \stackrel{def}{=} B.C'' \quad C'' \stackrel{def}{=} c.C$$

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:

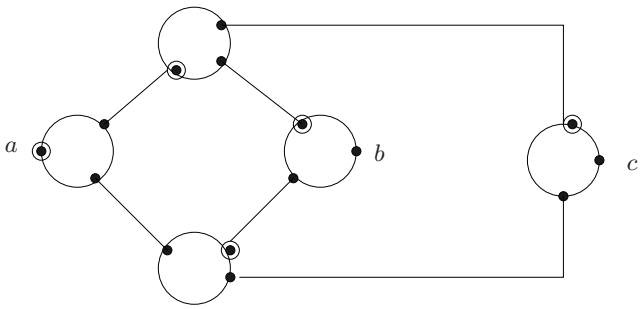
$$C_i \stackrel{def}{=} C[f_i] \qquad\qquad C' \stackrel{def}{=} C'_i[f_i] \quad C''_i \stackrel{def}{=} C''[f_i]$$

$$C^{(n)} \stackrel{def}{=} (C_1|C''_2|\ldots|C''_n)\backslash L$$

$$f_i = (a_1/a, b_{i+1}/b, \overline{b}_i/c)$$

$$L = \{b_1, \ldots, b_n\}$$

The system S in the flow graph below consists of five cyclers, each ready to fire at its ringed port:



(a) By considering S to be constructed from $C^{(4)}$ (the ring of 4 cyclers) – suitably relabelled - and one other cycler, prove that $S = M$, where $M \stackrel{def}{=} a.(b.c.M + c.b.M)$. ($M$ is a kind of multiplier; given an 'input' at a, it produces 'output' at both b and c, in arbitrary order. Many interesting agents can be built from cyclers and static combinators alone.)

(b) Write S first as a restricted Composition of five relabelled cyclers. Then convert it, using the static laws, into the form

$$(C^{(4)}[g]|C[f])\backslash L$$

The importance of this exercise is that it is often a good idea to consider a complex system as composed of subsystems, whose behaviour may be analysed and represented in a simple form before the whole system is considered. It is possible, but a bit tedious, to show that $S = M$ directly in the exercise, i.e. presenting S as a restricted Composition of five cyclers, rather than as a Composition of $C^{(4)}$ and one further cycler. The static laws, which tell us that different ways (as as the above) of building system from subsystems are indeed equivalent.

4. Use the equational laws of CCS to prove the following. State the laws you use.

   (a) $\alpha.(P + \tau.\tau.P) = \alpha.P$

   (b) $\tau.(P + \alpha.(Q + \tau.R)) = \tau.(P + \alpha.(Q + \tau.R)) + \alpha.R$

   (c) $a.\tau.b.\mathbf{0} + a.(b.\mathbf{0} + c.\mathbf{0} + \tau.b.\mathbf{0}) = a.(c.\mathbf{0} + \tau.b.\mathbf{0}) + a.(b.\mathbf{0} + c.\mathbf{0} + \tau.b.\mathbf{0})$

   (d) if $A \stackrel{def}{=} a.b.A$ and $B \stackrel{def}{=} b.a.B$ then $b.A = B$

5. Stating the laws you use, prove $Sys = Spec$, where

$$\begin{array}{lcl} Sys & \stackrel{def}{=} & (S|M|R)\backslash\{a,b,c\} \\ S & \stackrel{def}{=} & a.\overline{s}.\overline{b}.S \\ M & \stackrel{def}{=} & \overline{a}.b.\overline{c}.M \\ R & \stackrel{def}{=} & c.r.R \\ Spec & \stackrel{def}{=} & \tau.\overline{s}.Spec_1 \\ Spec_1 & \stackrel{def}{=} & r.Spec + \tau.Spec_2 \\ Spec_2 & \stackrel{def}{=} & r.\overline{s}.Spec_1 + \overline{s}.r.Spec_1 \end{array}$$

   Hint: use the Concurrency Workbench.

6. Explain why $a.(P + Q) = a.P + a.Q$ is not a law of CCS.

7. Prove that the following agents are equal. Mention any laws you use.

$$a.(c.\mathbf{0} + \tau.(\tau.b.\mathbf{0} + c.\mathbf{0})) + a.(b.\tau.\mathbf{0} + \tau.(b.\mathbf{0} + c.\mathbf{0}))$$
$$a.(b.\mathbf{0} + c.\tau.\mathbf{0}) + a.b.\mathbf{0} + a.(\tau.b.\mathbf{0} + b.\mathbf{0} + \tau.(\tau.b.\mathbf{0} + c.\mathbf{0}))$$

8. Let agents P, Q and R be as follows:

$$\begin{array}{lcl} P & \stackrel{def}{=} & a.\overline{d}.b.\overline{f}.\mathbf{0} \\ Q & \stackrel{def}{=} & a.\overline{e}.f.c.\mathbf{0} \\ R & \stackrel{def}{=} & d.e.\mathbf{0} \end{array}$$

Use the Expansion Law, and other laws of CCS as necessary, to obtain a new agent T such that $T = (P|Q|R)\backslash\{d, e, f\}$ and T uses only $\mathbf{0}$, prefixing with visible actions and sum combinators, (and no prefixing with τ).

9. (a) Explain how the following family of agent expressions encodes a counter storing a natural number:

$$\begin{aligned} C_0 &\stackrel{def}{=} up.C_1 \\ C_1 &\stackrel{def}{=} up.C_2 + down.C_0 \\ &\dots \\ C_n &\stackrel{def}{=} up.C_{n+1} + down.C_{n-1} \\ &\dots \end{aligned}$$

(b) Is the following agent C an implementation of the counter storing a natural number, i.e. is $C_0 = C$ valid? Justify formally your answer.

$$\begin{aligned} C &\stackrel{def}{=} up.(C|D) \\ D &\stackrel{def}{=} down.\mathbf{0} \end{aligned}$$

Hint: Use the constants law, unique solution law, Expansion Law and some other CCS laws to derive a set of equations for C and its up^n-derivatives, where up^n is the sequence on n actions up.