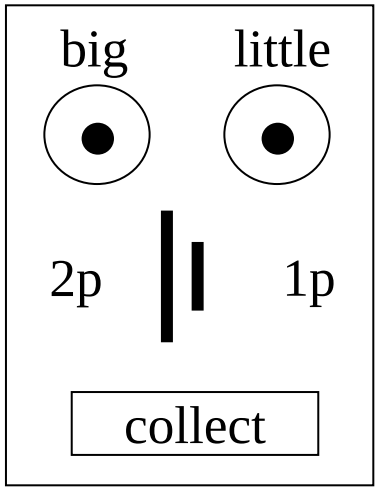
π Calculus

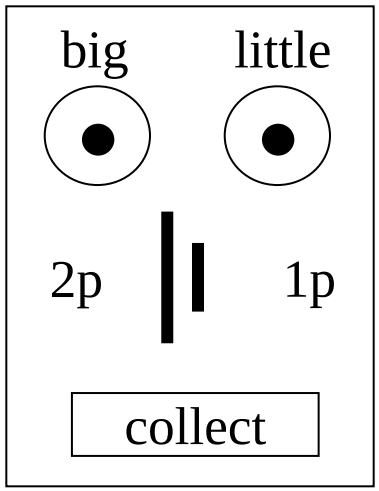
Reasoning about concurrency and communication (Part 2).

A Process with Alternative Behavior

A vending machine that dispenses chocolate candies allows either a 1p (p for pence) or a 2p coin to be inserted. After inserting a 1p coin, a button labelled “little” may be pressed and the machine will then dispense a small chocolate. After inserting a 2p coin, the “big” button may be pressed and the machine will then dispense a large chocolate. The candy must be collected before additional coins can be inserted.



An Process with Alternative Behavior



VM(big, little, collect, 1p, 2p) =

2p.big.$\overline{\text{collect}}$ largeChoc.VM(big, little, collect, 1p, 2p)

+ 1p.little.$\overline{\text{collect}}$ smallChoc.VM(big, little, collect, 1p, 2p)

The plus ("+") operator expresses alternative behavior.

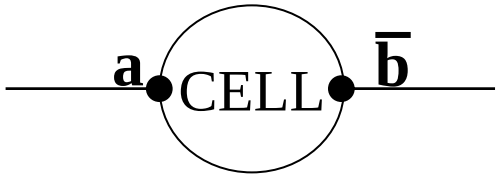
Modeling a Bounded Buffer

Suppose that a buffer has get and put operations and can hold up to three data items. Ignoring the content of the data items, and focusing only on the operations, a buffer can be defined as:

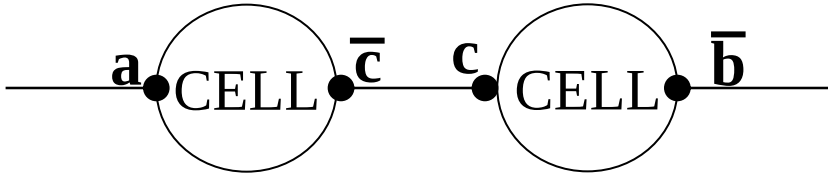
$$\text{Buffer}_0(\text{put}, \text{get}) = \text{put}.\text{Buffer}_1(\text{put}, \text{get})$$
$$\text{Buffer}_1(\text{put}, \text{get}) = \text{put}.\text{Buffer}_2(\text{put}, \text{get}) + \overline{\text{get}}.\text{Buffer}_0(\text{put}, \text{get})$$
$$\text{Buffer}_2(\text{put}, \text{get}) = \text{put}.\text{Buffer}_3(\text{put}, \text{get}) + \overline{\text{get}}.\text{Buffer}_1(\text{put}, \text{get})$$
$$\text{Buffer}_3(\text{put}, \text{get}) = \overline{\text{get}}.\text{Buffer}_2(\text{put}, \text{get})$$

Notice that this captures the idea that a get operation is not possible when the buffer is empty (i.e., in state Buffer_0) and a put operation is not possible when the buffer is full (i.e., in state Buffer_3).

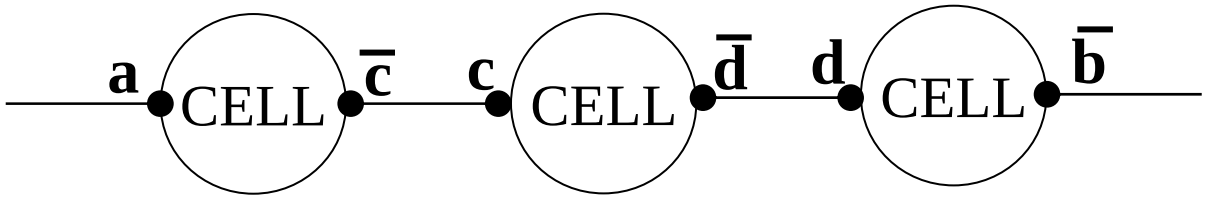
Reusing a Process Definition



$CELL(a,b) = a.\bar{b}.CELL(a,b)$



C0 = CELL(a, c)
C1 = CELL(c, b)
BUFF2 = (ν c) (C0 | C1)



C0 = CELL (a,c)
C1 = CELL (c,d)
C2 = CELL (d,b)
BUFF3 = (ν c)(ν d)(C0 | C1 | C2)

Modeling Mutual Exclusion

A lock to control access to a critical region is modeled by:

$$\begin{aligned} &Lock(lock, unlock) \quad = lock.Locked(lock, unlock) \\ &Locked(lock, unlock) = unlock.Lock(lock, unlock) \end{aligned}$$

A generic process with a critical region follows the locking protocol is:

$$\begin{aligned} &Process(enter, exit, lock, unlock) \\ &\qquad = \overline{lock}.enter.exit.\overline{unlock}.Process(enter, exit, lock, unlock) \end{aligned}$$

A system of two processes is:

$$\begin{aligned} &Process_1 = Process\ (enter_1, exit_1, lock, unlock) \\ &Process_2 = Process\ (enter_2, exit_2, lock, unlock) \\ &MutexSystem = (\nu\ lock)\ (\nu\ unlock)\ (Process_1 \mid Process_2 \mid Lock\ ) \end{aligned}$$

Modeling Mutual Exclusion

A system of two processes is:

$Process_1 = Process\ (enter_1, exit_1, lock, unlock)$
$Process_2 = Process\ (enter_2, exit_2, lock, unlock)$

$MutexSystem = new\ lock, unlock\ (Process_1 \mid Process_2 \mid Lock\ )$

A "specification" for this system is:

$MutexSpec(enter_1, exit_1, enter_2, exit_2)$

$= enter_1.exit_1.MutexSpec(enter_1, exit_1, enter_2, exit_2\ )$

$+ enter_2.exit_2.MutexSpec(enter_1, exit_1\ , enter_2, exit_2\ )$

Modeling a Bounded Buffer

The Buffer equations might be thought of as the "specification" of the bounded buffer because it only refers to states of the buffer and not to any internal components or machinery to create these states.

An "implementation" of the bounded buffer is readily available by re-labeling the BUFF3 agent developed earlier

$$CELL = a.\bar{b}.CELL$$

$$C0 = CELL\ (put\ ,\ c)$$
$$C1 = CELL\ (c\ ,\ d)$$
$$C2 = CELL\ (d\ ,\ get)$$
$$BufferImpl = (\nu\ c)\ (\nu\ d)\ (\ C0\ |\ C1\ |\ C2\ )$$

Equality of Processes

We would like to know if two process have the same behavior (interchagable), or if an implementation has the behavior required by a given specification (conformance). For example:

is $Buffer_0$ = BufferImpl ?

is MutexSystem = MutexSpec ?

How do we tell if two behaviors are the same?

Structural Congruence

Two expressions are the same if one can be transformed to the other using these rules:

(1) change of bound names : $(\nu\ a)\ (a.P) = (\nu\ c)\ (c.P)$

(2) reordering of terms in summation: $a.P + b.Q = b.Q + a.P$

(3) $P \mid 0 = P, \quad P \mid Q = Q \mid P, \quad P \mid (Q \mid R) = (P \mid Q) \mid R$

(4) $(\nu\ x)\ (P \mid Q) = P \mid (\nu\ x)\ Q$ if x is not a free name in P,

$(\nu\ x)\ 0 = 0, \quad (\nu\ x)\ (\nu\ y)\ P = (\nu\ y)\ (\nu\ x)\ P$

Reaction Rules

An equation can be changed by the application of these rules that express the "reaction" of the system being described:

COMM: $(x(y).P + M) \mid \overline{x}\, z.Q + N) \rightarrow \{z/y\}P \mid Q$

PAR: $$\frac{P \rightarrow P'}{P \mid Q \rightarrow P' \mid Q}$$

RES: $$\frac{P \rightarrow P'}{(\nu\, x)\, P \rightarrow (\nu\, x)\, P'}$$

STRUCT: $$\frac{Q=P \quad P \rightarrow P' \quad P'=Q'}{Q \rightarrow Q'}$$

Reaction Rules

Processes: $A(a,c) = a.A'(a,c)$ $\quad B(c,b) = c.B'(c,b)$

$A'(a,c) = \overline{c}.A(a,c)$ $\quad B'(c,b) = \overline{b}.B(c,b)$

A system: $System = \nu\, c\, (A \mid B)$

Show: $\nu\, c\, (A' \mid B) \longrightarrow \nu\, c\, (A \mid B')$

by REACT: $\overline{c}.A \mid c.B' \longrightarrow A \mid B'$

by RES: $\nu\, c(c.\overline{A} \mid c.B') \longrightarrow \nu\, c\, (A \mid B')$

by definition: $\nu\, c\, (A' \mid B) \longrightarrow \nu\, c(A \mid B')$

Depicting an Agent's Behavior

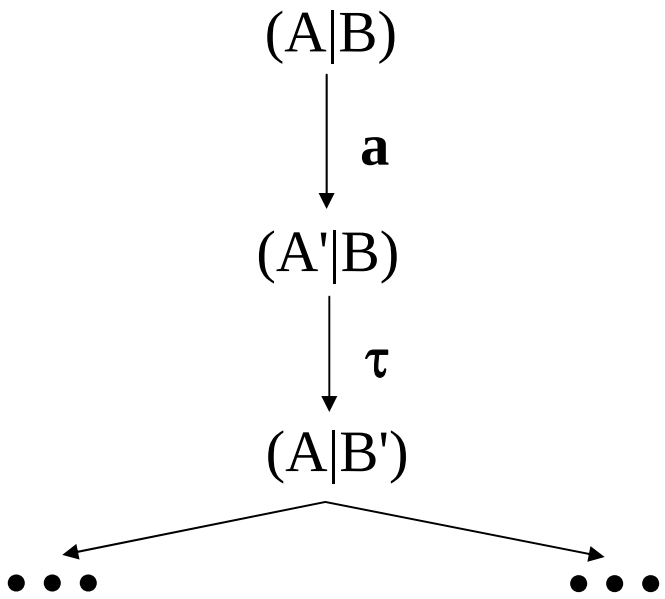
Define:

$A = a.A'$ $\quad\quad$ $B = c.B'$

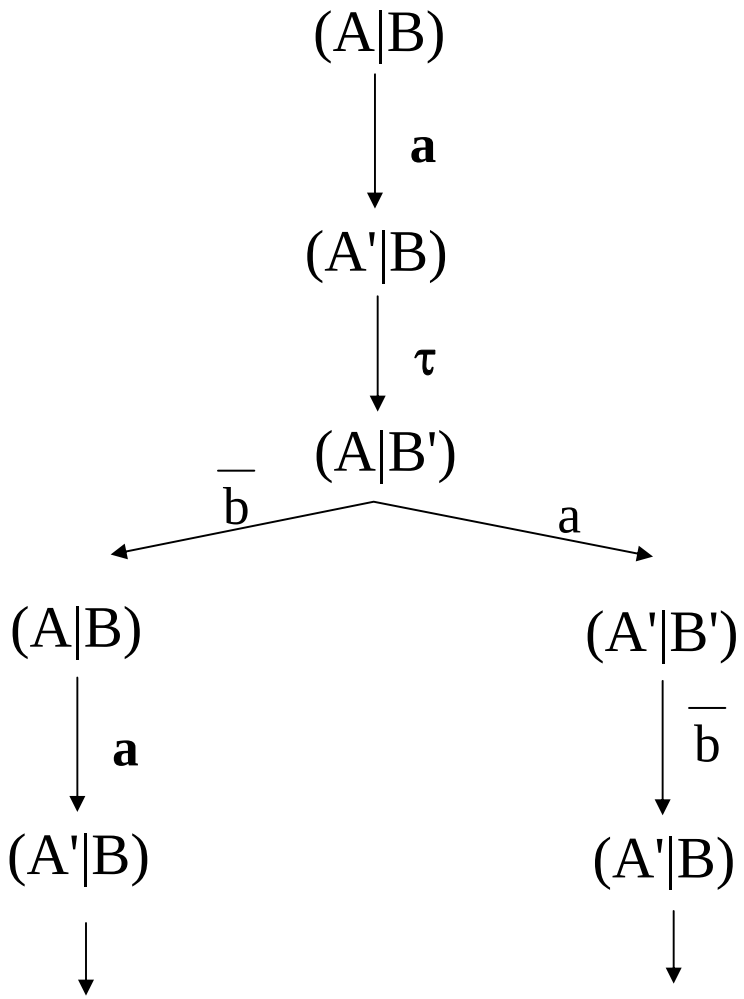
$A' = \overline{c}.A$ $\quad\quad$ $B' = \overline{b}.B$

System = $(\nu\ c)\ (\ A \mid B\ )$

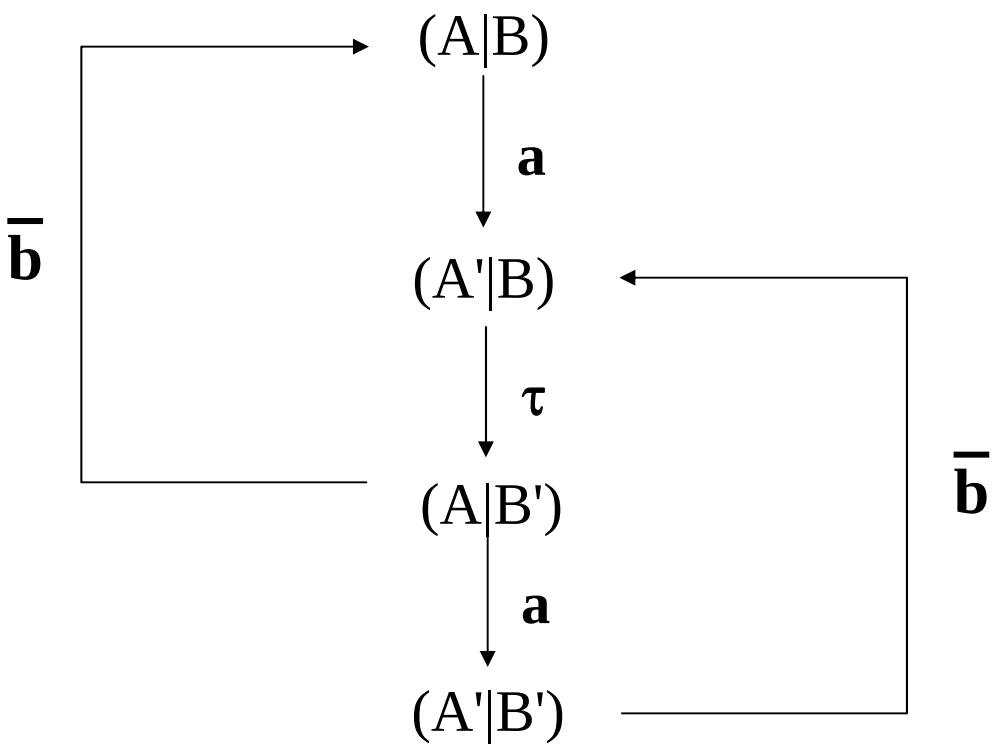
Draw a graph to show all possible sequences of actions. Here is the start:



More of the Behavior



Depicting an Agent's Behavior



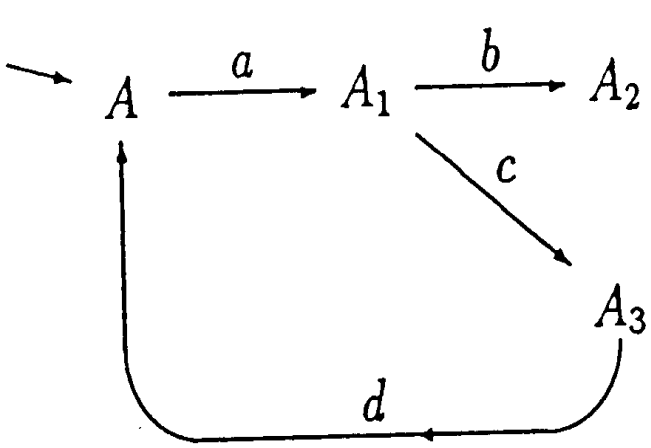
Equivalence of Agents

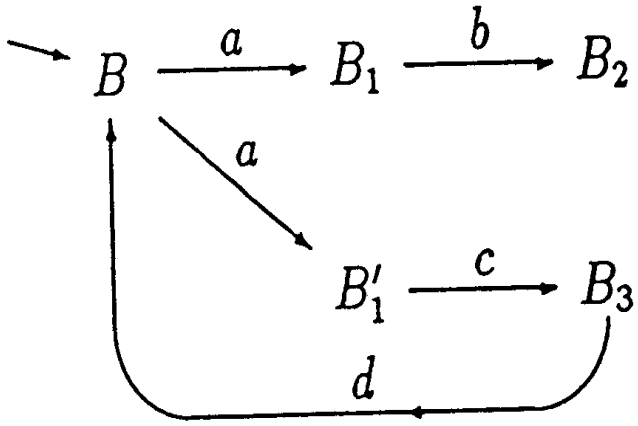
$$A \stackrel{\text{def}}{=} a.A_1 \qquad B \stackrel{\text{def}}{=} a.B_1 + a.B_1'$$

$$A_1 \stackrel{\text{def}}{=} b.A_2 + c.A_3 \qquad B_1 \stackrel{\text{def}}{=} b.B_2 \qquad B_1' \stackrel{\text{def}}{=} c.B_3$$

$$A_2 \stackrel{\text{def}}{=} 0 \qquad B_2 \stackrel{\text{def}}{=} 0$$

$$A_3 \stackrel{\text{def}}{=} d.A \qquad B_3 \stackrel{\text{def}}{=} d.B$$





Bisimulation

The behavior of two process are equal when each can simulate exactly the behavior of the other.