Introduction to Process Algebra

Based on:

Wan Fokkink, *Introduction to Process Algebra*, Springer-Verlag, 1999.

Reading list:

J. Baeten and P. Weijland, *Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

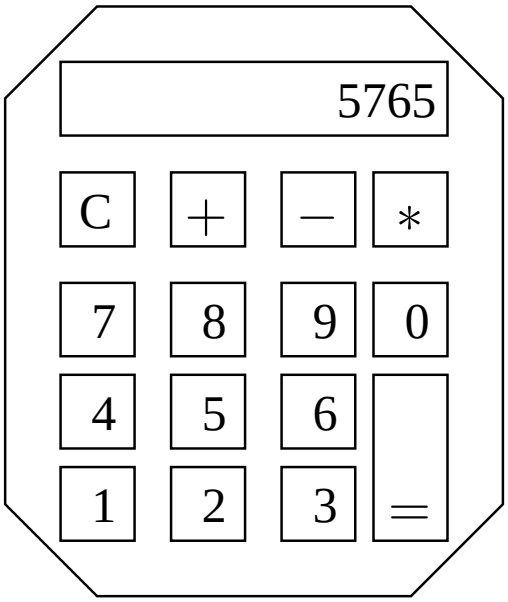
J. Baeten and C. Verhoef, Concrete process algebra, in *Handbook of Logic in Computer Science*, Volume IV, pp. 149–268, Oxford University Press, 1995.

R. Milner, *Communication and Concurrency*, Prentice Hall, 1989.

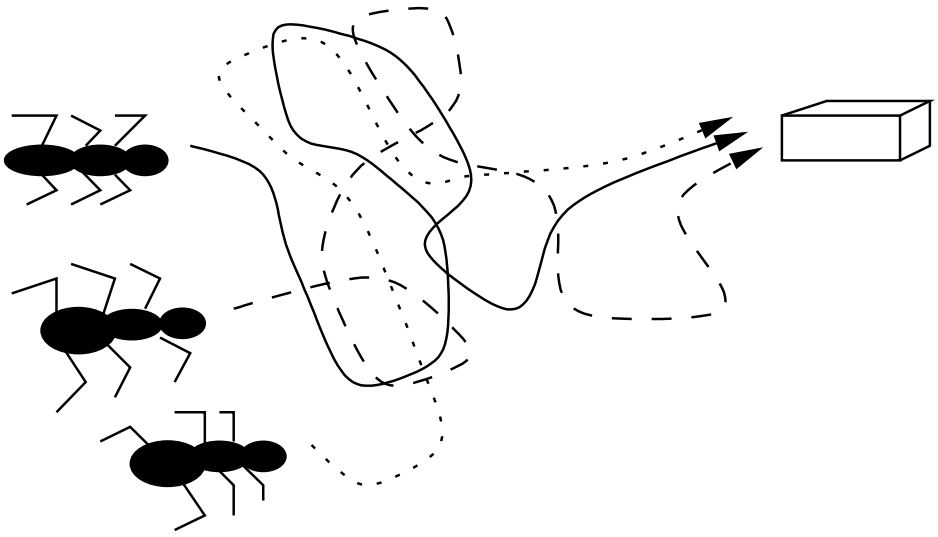
J. Baeten, ed., *Applications of Process Algebra*, Cambridge Tracts in Theoretical Computer Science 17, Cambridge University Press, 1990.

Examples of Concurrent Systems

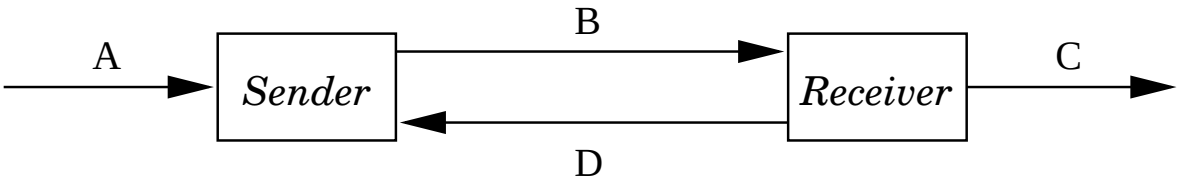
A pocket calculator:



A colony of ants:



The alternating bit protocol:



Data elements are sent from *Sender* to *Receiver* via faulty channel B. *Sender* alternatingly attaches bit 0 or bit 1 to data elements.

If *Receiver* receives a datum, it sends the attached bit to *Sender* via faulty channel D, to acknowledge reception. If *Receiver* receives a corrupted message, then it resends the preceding acknowledgement.

Sender keeps sending a datum with attached bit b until it receives acknowledgement b. Then it starts sending the next datum with attached bit $1 - b$ until it receives acknowledgement $1 - b$, et cetera.

Process Graphs

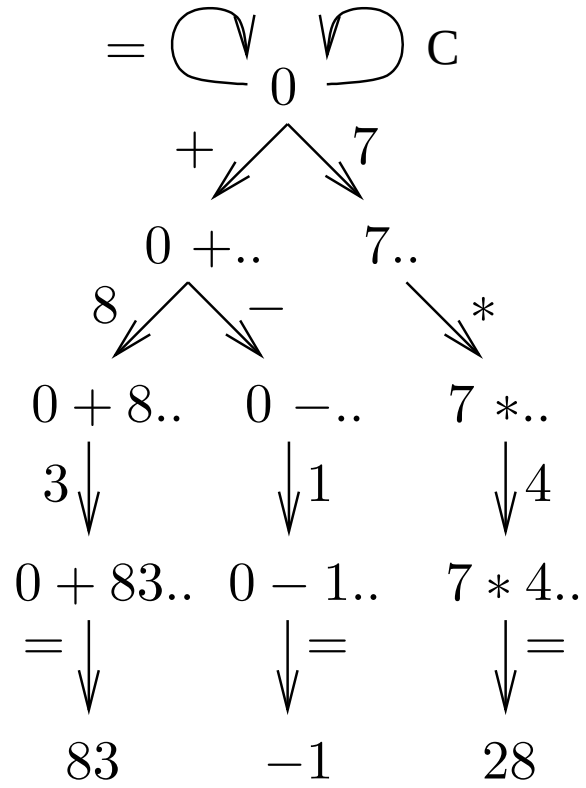
Assume a set S of *states*, together with a finite set A of *(atomic) actions*.

- A *transition* $s \xrightarrow{a} s'$ expresses that state s can evolve into state s' by execution of action a.
- A *transition* $s \xrightarrow{a} \surd$ expresses that state s can terminate successfully by execution of action a.

A *process (graph)* is a set of transitions, with a special *root state*.

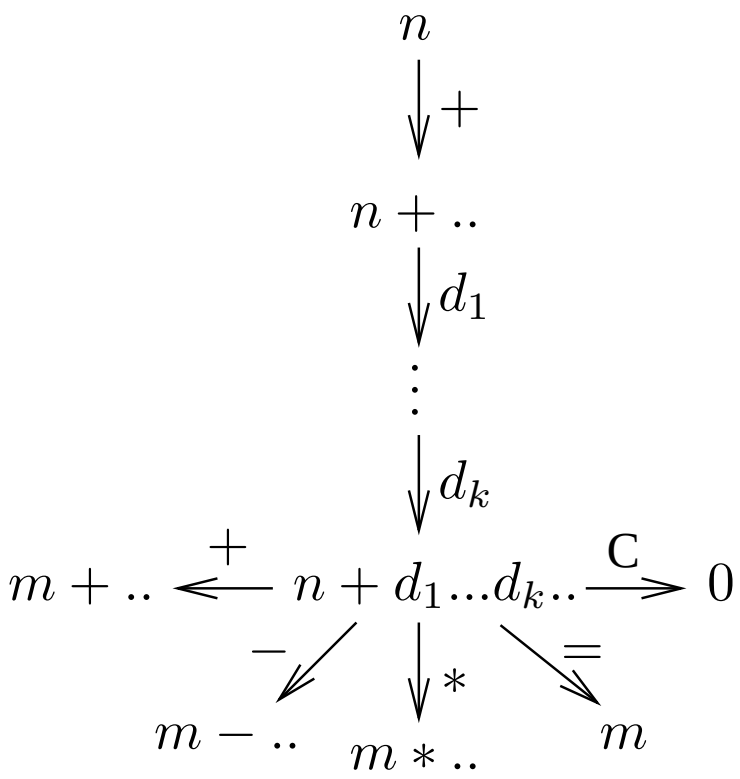
Example

Part of the process graph of the pocket calculator:



Example – continued

Behaviour of the plus button:



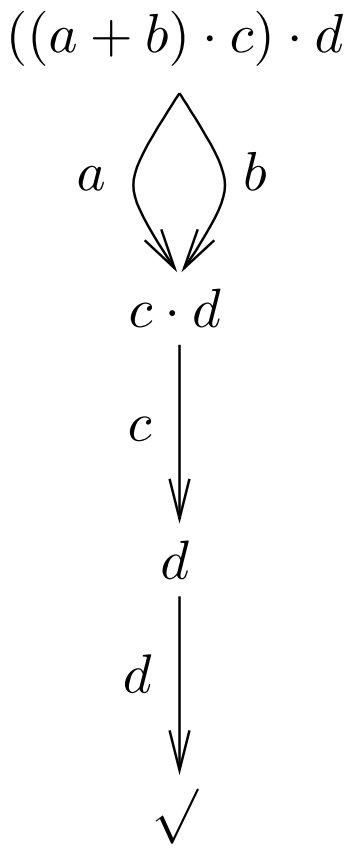
Basic Process Terms

Basic process terms are built from *atomic actions*, *alternative composition* and *sequential composition*.

- An *atomic action* a represents indivisible behaviour. It executes itself and then terminates successfully: $a \xrightarrow{a} \surd$.
- *Alternative composition*: the process term $t_1 + t_2$ executes the behaviour of either t_1 or t_2.
- *Sequential composition*: the process term $t_1 \cdot t_2$ first executes t_1, and upon successful termination proceeds to execute t_2.

Example

The basic process term $((a + b) \cdot c) \cdot d$
represents the process graph:



Transition Rules of Basic Process Algebra

$$\frac{}{v \xrightarrow{v} \surd}$$

$$\frac{x \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x'}{x + y \xrightarrow{v} x'}$$

$$\frac{y \xrightarrow{v} \surd}{x + y \xrightarrow{v} \surd} \qquad \frac{y \xrightarrow{v} y'}{x + y \xrightarrow{v} y'}$$

$$\frac{x \xrightarrow{v} \surd}{x \cdot y \xrightarrow{v} y} \qquad \frac{x \xrightarrow{v} x'}{x \cdot y \xrightarrow{v} x' \cdot y}$$

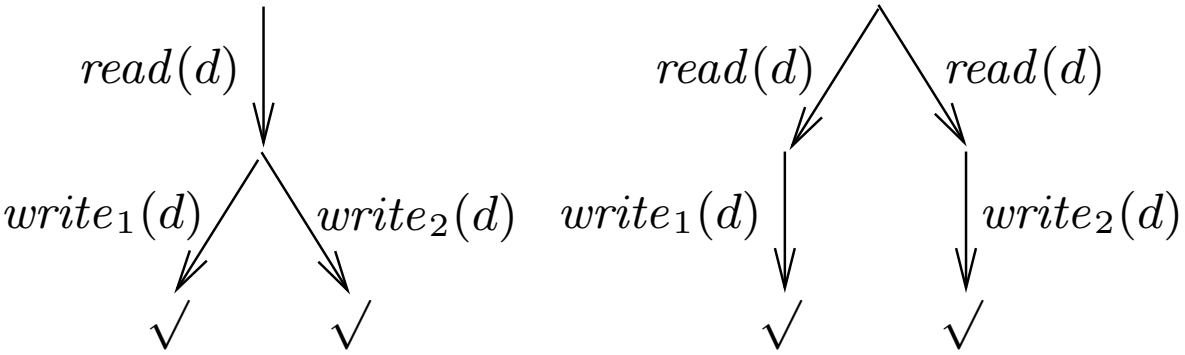
Proof of $((a+b)\cdot c)\cdot d \xrightarrow{b} c\cdot d$ from the transition rules:

$$\dfrac{\dfrac{\dfrac{b \xrightarrow{b} \surd}{a+b \xrightarrow{b} \surd}}{(a+b)\cdot c \xrightarrow{b} c}}{((a+b)\cdot c)\cdot d \xrightarrow{b} c\cdot d}$$

$$\dfrac{}{v \xrightarrow{v} \surd} \qquad \dfrac{y \xrightarrow{v} \surd}{x+y \xrightarrow{v} \surd} \qquad \dfrac{x \xrightarrow{v} \surd}{x\cdot y \xrightarrow{v} y} \qquad \dfrac{x \xrightarrow{v} x'}{x\cdot y \xrightarrow{v} x'\cdot y}$$

The transition rules at the right are instantiated in the corresponding proof steps.

Process equivalence based on traces is not satisfactory for our purposes. For example,



The first process reads datum d, and then decides whether it writes d on disc 1 or on disc 2. The second process makes a choice for disc 1 or disc 2 before it reads datum d.

Both processes display $read(d)write_1(d)$ and $read(d)write_2(d)$, so they are trace equivalent.

A crucial distinction between the two processes becomes apparent if disc 1 crashes. Then the first process always saves datum d on disc 2, while the second process may get stuck.

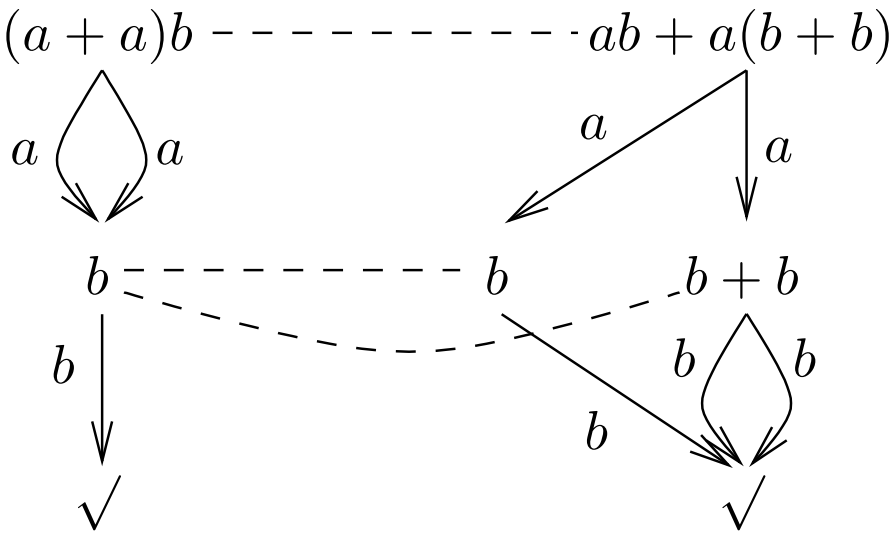
Bisimulation Equivalence

A *bisimulation* is a binary relation $\mathcal{B}$ on processes such that:

1. if $p \,\mathcal{B}\, q$ and $p \xrightarrow{a} p'$, then $q \xrightarrow{a} q'$ with $p' \,\mathcal{B}\, q'$
2. if $p \,\mathcal{B}\, q$ and $q \xrightarrow{a} q'$, then $p \xrightarrow{a} p'$ with $p' \,\mathcal{B}\, q'$
3. if $p \,\mathcal{B}\, q$ and $p \xrightarrow{a} \surd$, then $q \xrightarrow{a} \surd$
4. if $p \,\mathcal{B}\, q$ and $q \xrightarrow{a} \surd$, then $p \xrightarrow{a} \surd$

Two processes p and q are *bisimilar*, denoted $p \leftrightarrows q$, if there is a bisimulation relation $\mathcal{B}$ such that $p \,\mathcal{B}\, q$.

Example: $(a+a)b \;\underline{\leftrightarrow}\; ab + a(b+b)$



Bisimulation relation $\mathcal{B}$ defined by:

$(a+a)b \;\mathcal{B}\; ab + a(b+b)$

$b \;\mathcal{B}\; b$

$b \;\mathcal{B}\; b + b.$

Congruence

Bisimulation equivalence is a *congruence* over BPA.

That is, if $s \underline{\leftrightarrow} s'$ and $t \underline{\leftrightarrow} t'$, then
$s + t \underline{\leftrightarrow} s' + t'$ and $s \cdot t \underline{\leftrightarrow} s' \cdot t'$.

This congruence property follows from the fact that each transition rule ρ for BPA is in *panth* format:

- for each *premise* $t \xrightarrow{a} t'$ of ρ, the right-hand side t' is a single variable
- the *source* of ρ contains no more than one function symbol
- there are no multiple occurrences of the same variable in the right-hand sides of premises and in the source of ρ

Axioms for BPA Modulo Bisimulation

A1	$x + y$	$=$	$y + x$
A2	$(x + y) + z$	$=$	$x + (y + z)$
A3	$x + x$	$=$	x
A4	$(x + y) \cdot z$	$=$	$x \cdot z + y \cdot z$
A5	$(x \cdot y) \cdot z$	$=$	$x \cdot (y \cdot z)$

Equality Relation

The axioms induce an *equality* relation $=$ on basic process terms:

- (Substitution)

  If $s = t$ is an axiom and σ maps all variables in s and t to basic process terms, then $\sigma(s) = \sigma(t)$.

- (Equivalence)

  - $t = t$ for all basic process terms t
  - if $s = t$, then $t = s$
  - if $s = t$ and $t = u$, then $s = u$

- (Context)

  If $s = s'$ and $t = t'$, then $s + t = s' + t'$ and $s \cdot t = s' \cdot t'$.

Soundness: The axioms for BPA are *sound* modulo bisimulation equivalence.

That is, if $s = t$ then $s \underline{\leftrightarrow} t$.

Since bisimulation equivalence is a congruence over BPA, soundness of the axioms can be verified by checking that $\sigma(s) \underline{\leftrightarrow} \sigma(t)$ for each axiom $s = t$ and for all mappings σ from variables in s and t to basic process terms.

- A1: both $s + t$ and $t + s$ represent a choice between s and t
- A2: both $(s + t) + u$ and $s + (t + u)$ represent a choice between s, t and u
- A3: a choice between t and t amounts to a choice for t
- A4: both $(s + t) \cdot u$ and $s \cdot u + t \cdot u$ represent a choice between s and t, followed by u
- A5: both $(s \cdot t) \cdot u$ and $s \cdot (t \cdot u)$ represent s followed by t followed by u

Completeness: The axioms for BPA are *complete* modulo bisimulation equivalence.

That is, if $s \underline{\leftrightarrow} t$ then $s = t$.

Consider basic process terms *modulo AC of the* $+$: we write $s =_{\mathrm{AC}} t$ if s and t can be equated by axioms A1 and A2.

The axioms for BPA are turned into a *term rewriting system*, modulo AC of the $+$:

$$\begin{array}{rcl} x + y & =_{\mathrm{AC}} & y + x \\ (x + y) + z & =_{\mathrm{AC}} & x + (y + z) \\ x + x & \rightarrow & x \\ (x + y) \cdot z & \rightarrow & x \cdot z + y \cdot z \\ (x \cdot y) \cdot z & \rightarrow & x \cdot (y \cdot z) \end{array}$$

Each basic process term is reduced to a *normal form*, which does not reduce any further.

If $s \underline{\leftrightarrow} t$, and s and t have normal forms s' and t', respectively, then $s' =_{\mathrm{AC}} t'$. Hence,

$$s = s' =_{\mathrm{AC}} t' = t$$

Example:

$(a + a)(cd) + (bc)(d + d) = ((b + a)(c + c))d$

$$
\begin{aligned}
& (\underline{a + a})(cd) + (bc)(d + d) \\
\overset{\text{A3}}{\rightarrow} \quad & a(cd) + (bc)(\underline{d + d}) \\
\overset{\text{A3}}{\rightarrow} \quad & a(cd) + \underline{(bc)d} \\
\overset{\text{A5}}{\rightarrow} \quad & a(cd) + b(cd)
\end{aligned}
$$

$$
\begin{aligned}
& ((b + a)(\underline{c + c}))d \\
\overset{\text{A3}}{\rightarrow} \quad & \underline{((b + a)c)d} \\
\overset{\text{A5}}{\rightarrow} \quad & \underline{(b + a)(cd)} \\
\overset{\text{A4}}{\rightarrow} \quad & b(cd) + a(cd)
\end{aligned}
$$

The two resulting normal forms are equivalent modulo AC of the $+$, so the two original basic process terms are equal.

Communication

A *communication function* $\gamma : A \times A \to A$ produces for each pair of atomic actions a and b their communication $\gamma(a, b)$.

The communication function γ is *commutative* and *associative*:

$$\begin{aligned} \gamma(a, b) &\equiv \gamma(b, a) \\ \gamma(\gamma(a, b), c) &\equiv \gamma(a, \gamma(b, c)) \end{aligned}$$

Merge

The *merge* $\|$ executes the two process terms in its arguments in parallel:

$$\frac{x \xrightarrow{v} \surd}{x\|y \xrightarrow{v} y} \qquad \frac{x \xrightarrow{v} x'}{x\|y \xrightarrow{v} x'\|y}$$

$$\frac{y \xrightarrow{v} \surd}{x\|y \xrightarrow{v} x} \qquad \frac{y \xrightarrow{v} y'}{x\|y \xrightarrow{v} x\|y'}$$

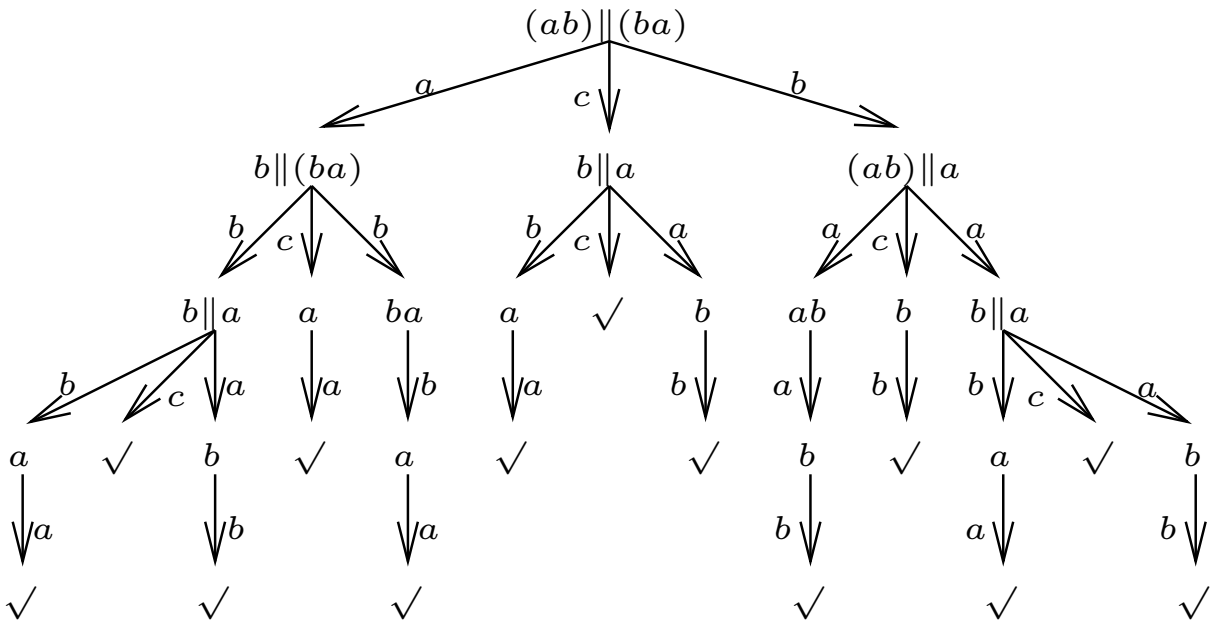
The merge can also execute a communication between initial transitions of its arguments:

$$\frac{x \xrightarrow{v} \surd \quad y \xrightarrow{w} \surd}{x\|y \xrightarrow{\gamma(v,w)} \surd} \qquad \frac{x \xrightarrow{v} \surd \quad y \xrightarrow{w} y'}{x\|y \xrightarrow{\gamma(v,w)} y'}$$

$$\frac{x \xrightarrow{v} x' \quad y \xrightarrow{w} \surd}{x\|y \xrightarrow{\gamma(v,w)} x'} \qquad \frac{x \xrightarrow{v} x' \quad y \xrightarrow{w} y'}{x\|y \xrightarrow{\gamma(v,w)} x'\|y'}$$

Example

The process graph of $(ab)\|(ba)$:



Left Merge and Communication Merge

Two auxiliary operators are needed to axiomatise the merge.

The *left merge* $\between$ executes an initial transition of its first argument:

$$\frac{x \xrightarrow{v} \surd}{x \between y \xrightarrow{v} y} \qquad \frac{x \xrightarrow{v} x'}{x \between y \xrightarrow{v} x' \| y}$$

The *communication merge* $|$ executes a communication between initial transitions of its arguments:

$$\frac{x \xrightarrow{v} \surd \quad y \xrightarrow{w} \surd}{x|y \xrightarrow{\gamma(v,w)} \surd} \qquad \frac{x \xrightarrow{v} \surd \quad y \xrightarrow{w} y'}{x|y \xrightarrow{\gamma(v,w)} y'}$$

$$\frac{x \xrightarrow{v} x' \quad y \xrightarrow{w} \surd}{x|y \xrightarrow{\gamma(v,w)} x'} \qquad \frac{x \xrightarrow{v} x' \quad y \xrightarrow{w} y'}{x|y \xrightarrow{\gamma(v,w)} x' \| y'}$$

Conservative Extension

PAP is a *conservative extension* of BPA, i.e., the process graphs of basic process terms remain the same.

This follows from the following two facts:

- The variables in each transition rule of BPA are *source-dependent*.
- The sources of the transition rules for merge, left merge and communication merge contain an occurrence of $\|$, $\between$ or $|$.

The *source-dependent* variables in a transition rule ρ are defined by:

- all variables in the *source* of ρ are source-dependent
- if $t \xrightarrow{a} t'$ is a premise of ρ and all variables in t are source-dependent, then all variables in t' are source-dependent

Example

The transition rule

$$\frac{x \xrightarrow{v} x'}{x \cdot y \xrightarrow{v} x' \cdot y}$$

is source-dependent.

Namely,

- the variables x and y in the source are source-dependent
- since x is source-dependent and there is a premise $x \xrightarrow{v} x'$, it follows that the variable x' is source-dependent

Congruence

Bisimulation equivalence is a *congruence* over PAP.

That is, if $s \underline{\leftrightarrow} s'$ and $t \underline{\leftrightarrow} t'$, then
$s + t \underline{\leftrightarrow} s' + t'$, $s \cdot t \underline{\leftrightarrow} s' \cdot t'$, $s \| t \underline{\leftrightarrow} s' \| t'$,
$s \between t \underline{\leftrightarrow} s' \between t'$ and $s | t \underline{\leftrightarrow} s' | t'$.

This congruence property follows from the fact that each transition rule ρ for BPA is in *panth* format:

- for each *premise* $t \stackrel{a}{\rightarrow} t'$ of ρ, the right-hand side t' is a single variable
- the *source* of ρ contains no more than one function symbol
- there are no multiple occurrences of the same variable in the right-hand sides of premises and in the source of ρ

Axioms for PAP

$$
\begin{array}{lrcl}
\text{M1} & x\|y & = & (x \between y + y \between x) + x|y \\
 & & & \\
\text{LM2} & v \between y & = & v \cdot y \\
\text{LM3} & (v \cdot x) \between y & = & v \cdot (x\|y) \\
\text{LM4} & (x+y) \between z & = & x \between z + y \between z \\
 & & & \\
\text{CM5} & v|w & = & \gamma(v,w) \\
\text{CM6} & v|(w \cdot y) & = & \gamma(v,w) \cdot y \\
\text{CM7} & (v \cdot x)|w & = & \gamma(v,w) \cdot x \\
\text{CM8} & (v \cdot x)|(w \cdot y) & = & \gamma(v,w) \cdot (x\|y) \\
\text{CM9} & (x+y)|z & = & x|z + y|z \\
\text{CM10} & x|(y+z) & = & x|y + x|z
\end{array}
$$

Soundness: The axioms for PAP are *sound* modulo bisimulation equivalence:

$$s = t \;\Rightarrow\; s \leftrightarrow t$$

Since bisimulation equivalence is a congruence over PAP, soundness of the axioms can be verified by checking that $\sigma(s) \leftrightarrow \sigma(t)$ for each axiom $s = t$ and for all mappings σ from variables in s and t to process terms.

Completeness: The axioms for PAP are *complete* modulo bisimulation equivalence:

$$s \leftrightarrow t \;\Rightarrow\; s = t$$

This follows by turning the axioms for PAP into a *term rewriting system*, modulo AC of the $+$. Each process term over PAP is reduced to a *normal form*. If $s \leftrightarrow t$, and s and t have normal forms s' and t', respectively, then $s' =_{\mathrm{AC}} t'$. Hence,

$$s = s' =_{\mathrm{AC}} t' = t$$

Deadlock and Encapsulation

The *deadlock* δ does not display any behaviour. There is no transition rule for this constant.

The *encapsulation* operator ∂_H, with $H \subseteq A$, renames all actions from H in its argument into δ:

$$\frac{x \xrightarrow{v} \surd \quad (v \notin H)}{\partial_H(x) \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x' \quad (v \notin H)}{\partial_H(x) \xrightarrow{v} \partial_H(x')}$$

The domain of the communication function γ is extended with δ:

$$\gamma : A \times A \to A \cup \{\delta\}.$$

If a and b do not communicate, then $\gamma(a, b) \equiv \delta$.

Encapsulation operators enable to enforce actions into communication. For example, $\partial_{\{a,b\}}(a \| b)$ can only execute $\gamma(a, b)$ (under the condition that this communication is not δ).

Conservative Extension

ACP is a *conservative extension* of PAP, i.e., the process graphs of process terms over PAP remain the same.

This follows from the following two facts:

- The variables in each transition rule of PAP are *source-dependent*.
- The sources of the transition rules for encapsulation operators contain ∂_H.

Congruence

Bisimulation is a *congruence* over ACP.

That is, if $s \leftrightarrow s'$ and $t \leftrightarrow t'$, then $s + t \leftrightarrow s' + t'$, $s \cdot t \leftrightarrow s' \cdot t'$, $s \| t \leftrightarrow s' \| t'$, $s \between t \leftrightarrow s' \between t'$, $s | t \leftrightarrow s' | t'$ and $\partial_H(s) \leftrightarrow \partial_H(t)$.

This congruence property holds because each transition rule for ACP is in *panth* format.

Axioms for Deadlock and Encapsulation

A6	$x + \delta$	$=$	x	
A7	$\delta \cdot x$	$=$	δ	
D1	$\partial_H(v)$	$=$	v	$(v \notin H)$
D2	$\partial_H(v)$	$=$	δ	$(v \in H)$
D3	$\partial_H(\delta)$	$=$	δ	
D4	$\partial_H(x + y)$	$=$	$\partial_H(x) + \partial_H(y)$	
D5	$\partial_H(x \cdot y)$	$=$	$\partial_H(x) \cdot \partial_H(y)$	
LM11	$\delta \lfloor\!\lfloor x$	$=$	δ	
CM12	$\delta \mid x$	$=$	δ	
CM13	$x \mid \delta$	$=$	δ	

Soundness: The axioms for ACP are *sound* modulo bisimulation equivalence:

$$s = t \;\Rightarrow\; s \underline{\leftrightarrow} t$$

Since bisimulation equivalence is a congruence over ACP, soundness of the axioms can be verified by checking that $\sigma(s) \underline{\leftrightarrow} \sigma(t)$ for each axiom $s = t$ and for all mappings σ from variables in s and t to process terms.

Completeness: The axioms for ACP are *complete* modulo bisimulation equivalence:

$$s \underline{\leftrightarrow} t \;\Rightarrow\; s = t$$

This follows by turning the axioms for ACP into a *term rewriting system*, modulo AC of the $+$. Each process term over ACP is reduced to a *normal form*. If $s \underline{\leftrightarrow} t$, and s and t have normal forms s' and t', respectively, then $s' =_{\mathrm{AC}} t'$. Hence,

$$s = s' =_{\mathrm{AC}} t' = t$$

Example

Let $\gamma(a,b) \equiv c$ and $\gamma(a',b') \equiv c'$ be the only communications between actions.

$$(a + a') \| (b + b')$$

$$\overset{\text{M1}}{=}$$

$$(a + a') \between (b + b') + (b + b') \between (a + a')$$
$$+ (a + a') | (b + b')$$

$$\overset{\text{LM4, CM9,10}}{=}$$

$$a \between (b + b') + a' \between (b + b') + b \between (a + a') + b' \between (a + a')$$
$$+ a|b + a|b' + a'|b + a'|b'$$

$$\overset{\text{LM2, CM5}}{=}$$

$$a \cdot (b + b') + a' \cdot (b + b') + b \cdot (a + a') + b' \cdot (a + a')$$
$$+ c + \delta + \delta + c'$$

$$\overset{\text{A6}}{=}$$

$$a \cdot (b + b') + a' \cdot (b + b') + b \cdot (a + a') + b' \cdot (a + a')$$
$$+ c + c'$$

Example – continued

Let H denote $\{a, a', b, b'\}$.

$$\partial_H((a + a')\|(b + b'))$$

$$=$$

$$\partial_H(a \cdot (b + b') + a' \cdot (b + b') + b \cdot (a + a')$$

$$+ b' \cdot (a + a') + c + c')$$

$$\overset{\text{D1,2,4,5}}{=}$$

$$\delta \cdot \partial_H(b + b') + \delta \cdot \partial_H(b + b') + \delta \cdot \partial_H(a + a')$$

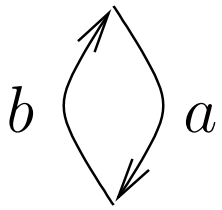
$$+ \delta \cdot \partial_H(a + a') + c + c'$$

$$\overset{\text{A6,7}}{=}$$

$$c + c'$$

Conclusion: ∂_H enforces communication between a and b and between a' and b'.

Recursion



Intuitively, this process can be captured by means of two *recursive equations*:

$$\begin{aligned} X &= aY \\ Y &= bX \end{aligned}$$

X and Y are *recursion variables*, representing the two states of the process.

A *recursive specification* is a collection

$$\begin{aligned} X_1 &= t_1(X_1, \ldots, X_n) \\ &\vdots \\ X_n &= t_n(X_1, \ldots, X_n) \end{aligned}$$

where the $t_i(X_1, \ldots, X_n)$ are process terms over ACP with possible occurrences of the recursion variables $X_1, \ldots, X_n$.

Solution

Processes $p_1, \ldots, p_n$ are a *solution* for a recursive specification

$$\{X_i = t_i(X_1, \ldots, X_n) \mid i \in \{1, \ldots, n\}\}$$

modulo bisimulation equivalence if $p_i \leftrightarrows t_i(p_1, \ldots, p_n)$ for $i \in \{1, \ldots, n\}$.

If $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$ are two solutions for the same recursive specification, then we want that $p_i \leftrightarrows q_i$ for $i \in \{1, \ldots, n\}$.

A recursive specification that allows more than one solution modulo bisimulation equivalence is

$$X \;=\; X$$

In contrast, the recursive specification

$$X \;=\; aX$$

has a unique solution modulo bisimulation.

Guardedness

A recursive specification

$$\{X_i = t_i(X_1, \ldots, X_n) \mid i \in \{1, \ldots, n\}\}$$

is *guarded* if the right-hand sides of its recursive equations can be adapted to the form

$$a_1 \cdot s_1(X_1, \ldots, X_n) + \cdots + a_k \cdot s_k(X_1, \ldots, X_n) + b_1 + \cdots + b_\ell$$

using the axioms of ACP and the possibility to replace recursion variables by the right-hand sides of their recursive equations.

Guarded recursive specifications are exactly the recursive specifications that have a unique solution modulo bisimulation equivalence.

Example: Let $\gamma(a,b) \equiv c$ and $\gamma(b,b) \equiv c$.

$\{X{=}Y \| Z,\ Y{=}Z + a,\ Z{=}bZ\}$ is guarded:

$Z{=}bZ$ is already in the desired form.

$Y{=}Z + a$ is brought in the desired form by replacing Z by the right-hand side bZ of its recursive equation.

$X{=}Y \| Z$ is brought in the desired form by replacing Y by $bZ + a$ and Z by bZ, and manipulating the resulting term $(bZ + a) \| bZ$ by the axioms:

$$
\begin{aligned}
& (bZ + a) \| bZ \\
= \ & (bZ + a) \between bZ + bZ \between (bZ + a) + (bZ + a) | bZ \\
= \ & bZ \between bZ + a \between bZ + bZ \between (bZ + a) + bZ | bZ + a | bZ \\
= \ & b(Z \| bZ) + abZ + b(Z \| (bZ + a)) + c(Z \| Z) + cZ
\end{aligned}
$$

Transition Rules for Recursion

Assume a guarded recursive specification E:

$$\{X_i = t_i(X_1, \ldots, X_n) \mid i \in \{1, \ldots, n\}\}$$

We introduce constants

$$\langle X_i|E\rangle \qquad\qquad (i \in \{1, \ldots, n\})$$

representing a solution for X_i in E.

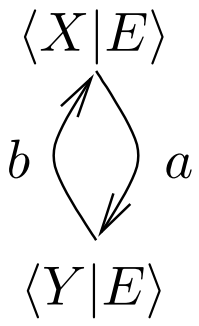
That is, $\langle X_i|E\rangle$ inherits the behaviour of $t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle)$:

$$\frac{t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle) \xrightarrow{v} \surd}{\langle X_i|E\rangle \xrightarrow{v} \surd}$$

$$\frac{t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle) \xrightarrow{v} y}{\langle X_i|E\rangle \xrightarrow{v} y}$$

Example

Let E denote $\{X{=}aY,\ Y{=}bX\}$. The process graph of $\langle X|E\rangle$ is:



We derive the transition $\langle X|E\rangle \xrightarrow{a} \langle Y|E\rangle$:

$$a \xrightarrow{a} \surd \qquad \frac{}{v \xrightarrow{v} \surd}$$

$$a \cdot \langle Y|E\rangle \xrightarrow{a} \langle Y|E\rangle \qquad \frac{x \xrightarrow{v} \surd}{x \cdot y \xrightarrow{v} y}$$

$$\langle X|E\rangle \xrightarrow{a} \langle Y|E\rangle \qquad \frac{a \cdot \langle Y|E\rangle \xrightarrow{v} y}{\langle X|E\rangle \xrightarrow{v} y}$$

The transition rules at the right are instantiated in the corresponding proof steps.

Conservative Extension

ACP with guarded recursion is a *conservative extension* of ACP.

This follows from the following two facts:

- The variables in each transition rule of ACP are *source-dependent*.
- The sources of the transition rules for guarded recursion contain $\langle X_i|E\rangle$.

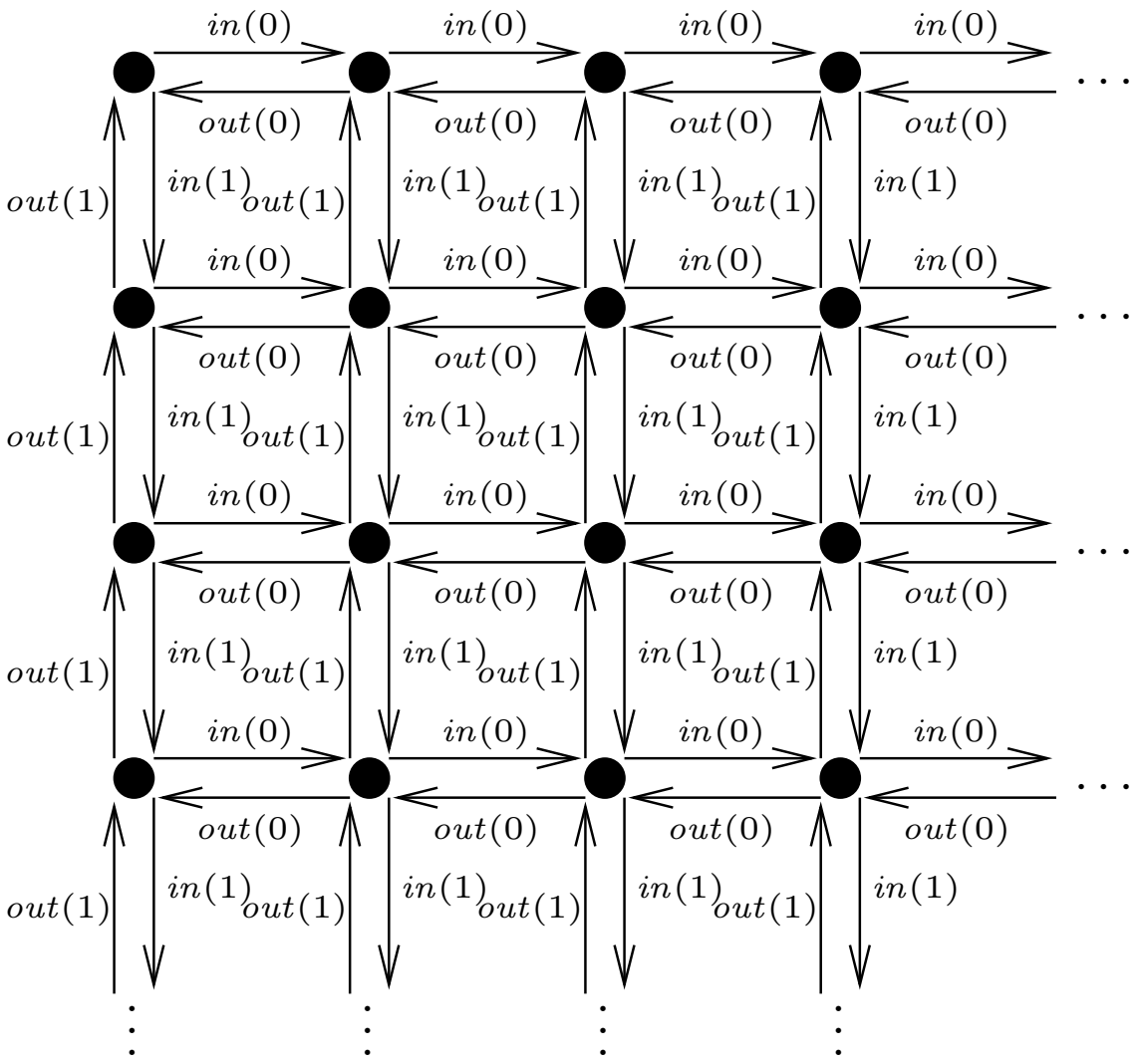
Congruence

Bisimulation equivalence is a *congruence* over ACP with guarded recursion.

This congruence property holds because each transition rule for ACP with guarded recursion is in *panth* format.

Example – bag over $\{0, 1\}$

We can put elements 0 and 1 into a bag, and subsequently collect these elements from the bag in arbitrary order.



The bag over $\{0, 1\}$ is specified by:

$$X \;=\; in(0)(X \| out(0)) + in(1)(X \| out(1))$$

Axioms for Recursion

Assume a guarded recursive specification E:

$$\{X_i = t_i(X_1, \ldots, X_n) \mid i \in \{1, \ldots, n\}\}$$

RDP $\quad \langle X_i|E\rangle = t_i(\langle X_1|E\rangle, \ldots, \langle X_n|E\rangle) \quad (i \in \{1, \ldots, n\})$

RSP $\quad$ If $y_i = t_i(y_1, \ldots, y_n)$ for $i \in \{1, \ldots, n\}$, then

$$y_i = \langle X_i|E\rangle \qquad (i \in \{1, \ldots, n\})$$

Soundness

The axioms for ACP with guarded recursion are *sound* modulo bisimulation equivalence:

$$s = t \;\Rightarrow\; s \mathrel{\underline{\leftrightarrow}} t$$

Since bisimulation equivalence is a congruence over ACP with guarded recursion, soundness can be verified by checking soundness of all instantiations of axioms.

RSP is not sound for *unguarded* recursion. For example, since $t = t$, RSP would yield

$$t \;=\; \langle X \,|\, X{=}X \rangle$$

for all process terms t.

Example

$$\begin{aligned}\langle Z \mid Z{=}aZ\rangle &\overset{\text{RDP}}{=} a\langle Z \mid Z{=}aZ\rangle\\ &\overset{\text{RDP}}{=} a(a\langle Z \mid Z{=}aZ\rangle)\\ &\overset{\text{A5}}{=} (aa)\langle Z \mid Z{=}aZ\rangle\end{aligned}$$

So by RSP,

$$\langle Z \mid Z{=}aZ\rangle \;=\; \langle X \mid X{=}(aa)X\rangle$$

Furthermore,

$$\begin{aligned}\langle Z \mid Z{=}aZ\rangle &\overset{\text{RDP}}{=} a\langle Z \mid Z{=}aZ\rangle\\ &\overset{\text{RDP}}{=} a(a\langle Z \mid Z{=}aZ\rangle)\\ &\overset{\text{RDP}}{=} a(a(a\langle Z \mid Z{=}aZ\rangle))\\ &\overset{\text{A5}}{=} ((aa)a)\langle Z \mid Z{=}aZ\rangle\end{aligned}$$

So by RSP,

$$\langle Z \mid Z{=}aZ\rangle \;=\; \langle Y \mid Y{=}((aa)a)Y\rangle$$

Hence,

$$\begin{aligned}\langle X \mid X{=}(aa)X\rangle &= \langle Z \mid Z{=}aZ\rangle\\ &= \langle Y \mid Y{=}((aa)a)Y\rangle\end{aligned}$$

Example: Let $\gamma(a,b) \equiv c$.

$$
\begin{array}{ll}
 & \langle X \mid X{=}aX \rangle \| \langle Y \mid Y{=}bY \rangle \\
= & \langle X \mid X{=}aX \rangle \between \langle Y \mid Y{=}bY \rangle \\
+ & \langle Y \mid Y{=}bY \rangle \between \langle X \mid X{=}aX \rangle \\
+ & \langle X \mid X{=}aX \rangle | \langle Y \mid Y{=}bY \rangle \\
= & a(\langle X \mid X{=}aX \rangle \| \langle Y \mid Y{=}bY \rangle) \\
+ & b(\langle Y \mid Y{=}bY \rangle \| \langle X \mid X{=}aX \rangle) \\
+ & c(\langle X \mid X{=}aX \rangle \| \langle Y \mid Y{=}bY \rangle)
\end{array}
$$

Hence,

$$
\begin{array}{ll}
 & \partial_{\{a,b\}}(\langle X \mid X{=}aX \rangle \| \langle Y \mid Y{=}bY \rangle) \\
= & c \cdot \partial_{\{a,b\}}(\langle X \mid X{=}aX \rangle \| \langle Y \mid Y{=}bY \rangle)
\end{array}
$$

So by RSP,

$$\partial_{\{a,b\}}(\langle X \mid X{=}aX \rangle \| \langle Y \mid Y{=}bY \rangle) = \langle Z \mid Z{=}cZ \rangle$$

Let E, E' be guarded recursive specifications, where E' is obtained from E by adapting the right-hand sides of its recursive equations,

- using the axioms for ACP with guarded recursion, and
- replacing recursion variables by the right-hand sides of their recursive equations.

Then $\langle X|E\rangle = \langle X|E'\rangle$ can be derived from the axioms for ACP with guarded recursion for all recursion variables X in E.

Example

$$\langle X \mid X{=}aX{+}aX\rangle = \langle X \mid X{=}(aa)X\rangle$$

can be derived by

- first applying A3 ($x + x = x$),
- next replacing X by its right-hand side aX,
- and finally applying A5 ($(xy)z = x(yz)$).

Regular Processes

A process p is *regular* if there are only finitely many processes p' such that $p \xrightarrow{a} \cdots \xrightarrow{b} p'$.

A recursive specification is *linear* if its recursive equations are of the form

$$X \;=\; a_1 \cdot X_1 + \cdots + a_k \cdot X_k + b_1 + \cdots + b_\ell$$

Each regular process can be described by a linear recursive specification, in which for each state s there is a recursion variable X_s:

- If state s can evolve into state s' by execution of a, then there is a summand $aX_{s'}$ in the recursive equation for X_s.
- If state s can terminate successfully by execution of a, then there is a summand a in the recursive equation for X_s.

Vice versa, a linear recursive specification gives rise to a regular process.

Completeness for Regular Processes

The axioms for ACP with *linear* recursion are *complete* modulo bisimulation equivalence:

$$s \underline{\leftrightarrow} t \;\Rightarrow\; s = t$$

This follows from two facts:

- Each process term over ACP with linear recursion is equal to a term $\langle X|E\rangle$ with E a linear recursive specification.
- If $\langle X|E\rangle \underline{\leftrightarrow} \langle Y|E'\rangle$ for linear recursive specifications E, E', then $\langle X|E\rangle = \langle Y|E'\rangle$.

Hence, if $s \underline{\leftrightarrow} t$, then

$$s = \langle X|E\rangle = \langle Y|E'\rangle = t$$

The axioms are *not* complete for ACP with *guarded* recursion modulo bisimulation.

Projection Operators

The *projection* operator π_n for $n \in \mathbb{N}$ executes all transitions of its argument up to depth n:

$$\frac{x \xrightarrow{v} \surd}{\pi_{n+1}(x) \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x'}{\pi_{n+1}(x) \xrightarrow{v} \pi_n(x')}$$

Conservative Extension: ACP with guarded recursion and projection operators is a *conservative extension* of ACP with guarded recursion.

Congruence: Bisimulation equivalence is a *congruence* over ACP with guarded recursion and projection operators.

Axioms:

PR1	$\pi_n(x+y)$	$=$	$\pi_n(x)+\pi_n(y)$
PR2	$\pi_{n+1}(v)$	$=$	v
PR3	$\pi_{n+1}(v \cdot x)$	$=$	$v \cdot \pi_n(x)$
PR4	$\pi_0(x)$	$=$	δ
PR5	$\pi_n(\delta)$	$=$	δ

Approximation Induction Principle

AIP If $\pi_n(x) = \pi_n(y)$ for $n \in \mathbb{N}$, then $x = y$

Soundness: The axioms for ACP with guarded recursion and projection operators are *sound* modulo bisimulation equivalence:

$$s = t \;\Rightarrow\; s \underline{\leftrightarrow} t$$

Strength of AIP: For each pair of bisimilar process terms s and t in ACP with guarded recursion, $\pi_n(s) = \pi_n(t)$ for $n \in \mathbb{N}$ can be derived from the axioms for ACP with guarded recursion and projection operators.

Example: We derive by induction on n:

$$\begin{aligned} & \pi_n(\langle Y \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^k) \\ = \quad & \pi_n(\langle X \mid X{=}aXb{+}b\rangle \cdot b^k) \\ = \quad & \pi_n(\langle Z \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^k) \end{aligned}$$

The case $n = 0$ is trivial. By induction:

$$\begin{aligned} & \pi_{n+1}(\langle X \mid X{=}aXb{+}b\rangle \cdot b^k) \\ = \ & \pi_{n+1}(a \cdot \langle X \mid X{=}aXb{+}b\rangle \cdot b^{k+1} + b^{k+1}) \\ = \ & a \cdot \pi_n(\langle X \mid X{=}aXb{+}b\rangle \cdot b^{k+1}) + \pi_{n+1}(b^{k+1}) \\ = \ & a \cdot \pi_n(\langle Z \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^{k+1}) \\ & \qquad + \pi_{n+1}(b^{k+1}) \\ = \ & \pi_{n+1}(a \cdot \langle Z \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^{k+1} + b^{k+1}) \\ = \ & \pi_{n+1}(\langle Y \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^k) \end{aligned}$$

Likewise,

$$\begin{aligned} & \pi_{n+1}(\langle X \mid X{=}aXb{+}b\rangle \cdot b^k) \\ = \quad & \pi_{n+1}(\langle Z \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^k) \end{aligned}$$

So by AIP,

$$\begin{aligned} & \langle Y \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^k \\ = \quad & \langle Z \mid Y{=}aZb{+}b, Z{=}aYb{+}b\rangle \cdot b^k \end{aligned}$$

Abstraction

The *silent step* τ represents an internal action:

$$\frac{}{\tau \xrightarrow{\tau} \surd}$$

The *abstraction* operator τ_I, with $I \subseteq A$, renames all actions from I in its argument to τ:

$$\frac{x \xrightarrow{v} \surd \quad (v \notin I)}{\tau_I(x) \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x' \quad (v \notin I)}{\tau_I(x) \xrightarrow{v} \tau_I(x')}$$

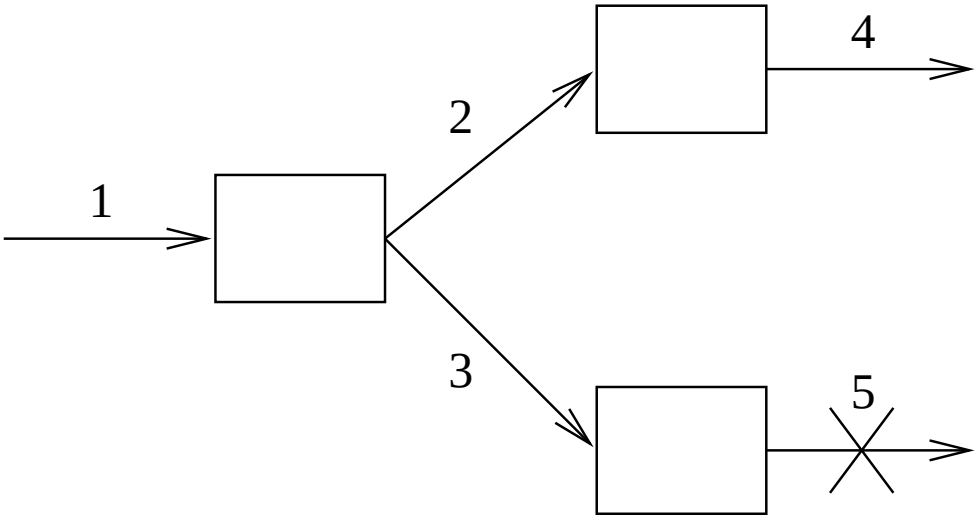
$$\frac{x \xrightarrow{v} \surd \quad (v \in I)}{\tau_I(x) \xrightarrow{\tau} \surd} \qquad \frac{x \xrightarrow{v} x' \quad (v \in I)}{\tau_I(x) \xrightarrow{\tau} \tau_I(x')}$$

Branching Bisimulation Equivalence

Not all τ-transitions are silent.

Example – A malfunctioning channel.

A datum received via channel 1 is sent into channel 2 or 3. A datum communicated via channel 2 it is sent into channel 4. A datum communicated via channel 3 gets stuck, because the subsequent channel 5 is broken.



The system gets into a deadlock if a datum is transferred via channel 3. This deadlock should not disappear when abstracting away from the communication actions via channels 2 and 3.

$a + \tau\delta$ and a are not equivalent.

$\partial_{\{b\}}(a + \tau b)$ and $\partial_{\{b\}}(a + b)$ are not equivalent.

$a + \tau b$ and $a + b$ are not equivalent.

A correct answer to the question

which τ-transitions are truly silent?

turns out to be

those τ-transitions that do not lose possible behaviours!

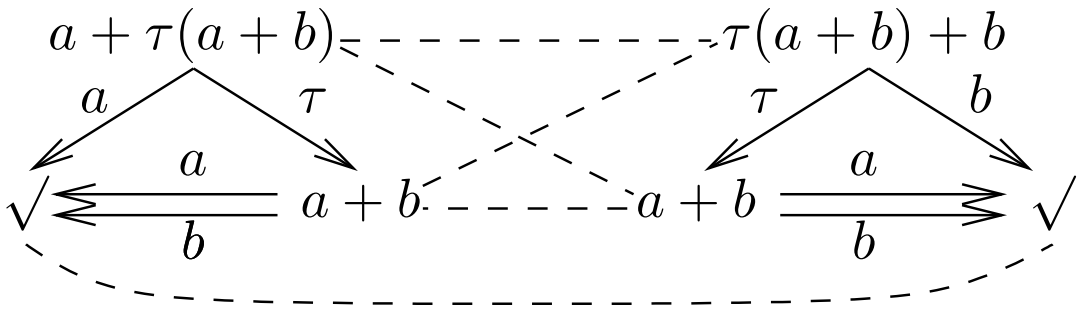
For example, $a + \tau(a + b)$ and $a + b$ are equivalent: after executing the τ in the first process term it is still possible to execute a.

Let $\surd \downarrow$. A *branching bisimulation* is a binary relation $\mathcal{B}$ on processes such that:

1. if $p \mathcal{B} q$ and $p \xrightarrow{a} p'$, then
   - either $a \equiv \tau$ and $p' \mathcal{B} q$
   - or $q \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_0$ such that $p \mathcal{B} q_0$ and $q_0 \xrightarrow{a} q'$ with $p' \mathcal{B} q'$
2. if $p \mathcal{B} q$ and $q \xrightarrow{a} q'$, then
   - either $a \equiv \tau$ and $p \mathcal{B} q'$
   - or $p \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_0$ such that $p_0 \mathcal{B} q$ and $p_0 \xrightarrow{a} p'$ with $p' \mathcal{B} q'$
3. if $p \mathcal{B} q$ and $p \downarrow$, then $q \xrightarrow{\tau} \cdots \xrightarrow{\tau} q_0$ such that $q_0 \downarrow$
4. if $p \mathcal{B} q$ and $q \downarrow$, then $p \xrightarrow{\tau} \cdots \xrightarrow{\tau} p_0$ such that $p_0 \downarrow$

Two processes p and q are *branching bisimilar*, denoted $p \underline{\leftrightarrow}_b q$, if there is a branching bisimulation relation $\mathcal{B}$ such that $p \mathcal{B} q$.

Example: $a + \tau(a+b) \underline{\leftrightarrow}_b \tau(a+b) + b$



Branching bisimulation relation $\mathcal{B}$ is defined by:

$a + \tau(a+b)\, \mathcal{B}\, \tau(a+b) + b$

$a + b\, \mathcal{B}\, \tau(a+b) + b$

$a + \tau(a+b)\, \mathcal{B}\, a + b$

$a + b\, \mathcal{B}\, a + b$

$\surd\, \mathcal{B}\, \surd$

Rooted Branching Bisimulation

$a + \tau\delta$ and a are not equivalent.

$\partial_{\{b\}}(a + \tau b)$ and $\partial_{\{b\}}(a + b)$ are not equivalent.

$a + \tau b$ and $a + b$ are not equivalent.

b and τb are not equivalent.

A *rooted branching bisimulation* is a binary relation $\mathcal{B}$ on processes such that:

1. if $p \mathcal{B} q$ and $p \xrightarrow{a} p'$, then $q \xrightarrow{a} q'$ with $p' \underline{\leftrightarrow}_b q'$
2. if $p \mathcal{B} q$ and $q \xrightarrow{a} q'$, then $p \xrightarrow{a} p'$ with $p' \underline{\leftrightarrow}_b q'$

Two processes p and q are *rooted branching bisimilar*, denoted $p \underline{\leftrightarrow}_{rb} q$, if there is a rooted branching bisimulation relation $\mathcal{B}$ such that $p \mathcal{B} q$.

Guarded Linear Recursion

All process terms τs are solutions for the recursive specification $X = \tau X$, because $\tau s \leftrightarrows_{rb} \tau\tau s$. Hence, $X = \tau X$ is *unguarded*.

A recursive specification is *linear* if its recursive equations are of the form

$$X \;=\; a_1 X_1 + \cdots + a_k X_k + b_1 + \cdots + b_\ell$$

A linear recursive specification E is *guarded* if there does not exist an infinite sequence of τ-transitions

$$\langle X|E\rangle \xrightarrow{\tau} \langle X'|E\rangle \xrightarrow{\tau} \langle X''|E\rangle \xrightarrow{\tau} \cdots$$

The guarded linear recursive specifications are exactly the linear recursive specifications that have a unique solution modulo rooted branching bisimulation.

Conservative Extension: ACP_τ with guarded linear recursion is a *conservative extension* of ACP with linear recursion.

This follows from the following three facts:

- The transition rules of ACP and for linear recursion are all source-dependent.
- The sources of the transition rules for the silent step and abstraction contain τ or τ_I. The sources of the transition rules for a guarded linear recursive specification E that includes a τ contain the fresh constant $\langle X|E\rangle$.
- Each transition rule for alternative composition, sequential composition or linear recursion that involves τ-transitions contains a premise that includes the fresh relation symbol $\xrightarrow{\tau}$ or predicate $\xrightarrow{\tau}\surd$, and a left-hand side of which all variables occur in the source of the transition rule.

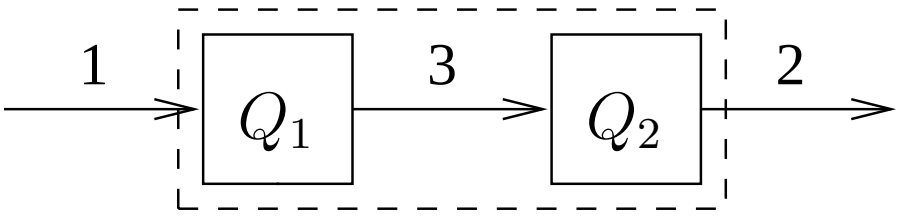
Congruence: Rooted branching bisimulation equivalence is a *congruence* over ACP_τ with guarded linear recursion.

Axioms for Abstraction

B1	$v \cdot \tau$	$=$	v	
B2	$v \cdot (\tau \cdot (x + y) + x)$	$=$	$v \cdot (x + y)$	
TI1	$\tau_I(v)$	$=$	v	$(v \notin I)$
TI2	$\tau_I(v)$	$=$	τ	$(v \in I)$
TI3	$\tau_I(\delta)$	$=$	δ	
TI4	$\tau_I(x + y)$	$=$	$\tau_I(x) + \tau_I(y)$	
TI5	$\tau_I(x \cdot y)$	$=$	$\tau_I(x) \cdot \tau_I(y)$	

Example – Queues

$$
\begin{aligned}
Q_1 &= r_1\, s_3\, Q_1 \\
Q_2 &= r_3\, s_2\, Q_2
\end{aligned}
$$



Buffers Q_2 and Q_1 of capacity one in parallel

$$\tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1))$$

behave as a buffer of capacity two:

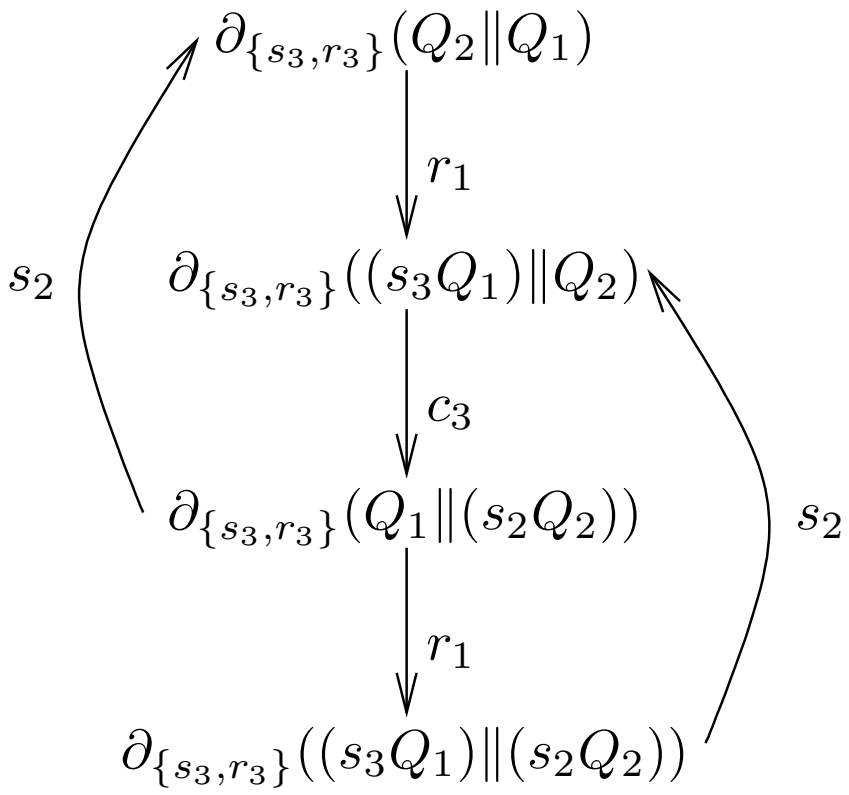
$$
\begin{aligned}
X &= r_1\, Y \\
Y &= r_1\, Z + s_2\, X \\
Z &= s_2\, Y
\end{aligned}
$$

$$
\begin{aligned}
& Q_2 \| Q_1 \\
= \quad & Q_2 \between Q_1 + Q_1 \between Q_2 + Q_2 | Q_1 \\
= \quad & (r_3 s_2 Q_2) \between Q_1 + (r_1 s_3 Q_1) \between Q_2 \\
& + (r_3 s_2 Q_2) | (r_1 s_3 Q_1) \\
= \quad & r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2) \\
& + \delta \cdot ((s_2 Q_2) \| (s_3 Q_1)) \\
= \quad & r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2)
\end{aligned}
$$

$$
\begin{aligned}
& \partial_{\{s_3, r_3\}} (Q_2 \| Q_1) \\
= \quad & \partial_{\{s_3, r_3\}} (r_3 \cdot ((s_2 Q_2) \| Q_1) + r_1 \cdot ((s_3 Q_1) \| Q_2)) \\
= \quad & \partial_{\{s_3, r_3\}} (r_3 \cdot ((s_2 Q_2) \| Q_1)) \\
& + \partial_{\{s_3, r_3\}} (r_1 \cdot ((s_3 Q_1) \| Q_2)) \\
= \quad & \partial_{\{s_3, r_3\}} (r_3) \cdot \partial_{\{s_3, r_3\}} ((s_2 Q_2) \| Q_1) \\
& + \partial_{\{s_3, r_3\}} (r_1) \cdot \partial_{\{s_3, r_3\}} ((s_3 Q_1) \| Q_2) \\
= \quad & \delta \cdot \partial_{\{s_3, r_3\}} ((s_2 Q_2) \| Q_1) \\
& + r_1 \cdot \partial_{\{s_3, r_3\}} ((s_3 Q_1) \| Q_2) \\
= \quad & r_1 \cdot \partial_{\{s_3, r_3\}} ((s_3 Q_1) \| Q_2)
\end{aligned}
$$

Likewise we can derive:

$$\partial_{\{s_3,r_3\}}((s_3Q_1)\|Q_2) = c_3 \cdot \partial_{\{s_3,r_3\}}(Q_1\|(s_2Q_2))$$
$$\partial_{\{s_3,r_3\}}(Q_1\|(s_2Q_2)) = r_1 \cdot \partial_{\{s_3,r_3\}}((s_3Q_1)\|(s_2Q_2)) + s_2 \cdot \partial_{\{s_3,r_3\}}(Q_2\|Q_1)$$
$$\partial_{\{s_3,r_3\}}((s_3Q_1)\|(s_2Q_2)) = s_2 \cdot \partial_{\{s_3,r_3\}}((s_3Q_1)\|Q_2)$$



$$
\begin{aligned}
& \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
= \quad & \tau_{\{c_3\}}(r_1 \cdot \partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2)) \\
= \quad & r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| Q_2)) \\
= \quad & r_1 \cdot \tau_{\{c_3\}}(c_3 \cdot \partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
= \quad & r_1 \cdot \tau \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
= \quad & r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)))
\end{aligned}
$$

Likewise we can derive

$$
\begin{aligned}
& \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \; = \\
& \qquad r_1 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2))) \\
& \qquad + s_2 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
& \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2))) \; = \\
& \qquad s_2 \cdot \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2)))
\end{aligned}
$$

Hence, a solution for the linear recursive specification E for the buffer of capacity two is

$$
\begin{aligned}
X \quad & := \quad \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_2 \| Q_1)) \\
Y \quad & := \quad \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}(Q_1 \| (s_2 Q_2))) \\
Z \quad & := \quad \tau_{\{c_3\}}(\partial_{\{s_3,r_3\}}((s_3 Q_1) \| (s_2 Q_2)))
\end{aligned}
$$

Cluster Fair Abstraction Rule

τ-loops can be constructed by abstraction: $\tau_{\{a\}}(\langle X \mid X{=}aX\rangle)$ executes τ's until infinity.

Let E be a guarded linear recursive specification, and $I \subseteq A$.

Recursion variables X and Y in E are in the same *cluster* for I if $\langle X|E\rangle \stackrel{b_1}{\rightarrow} \cdots \stackrel{b_m}{\rightarrow} \langle Y|E\rangle$ and $\langle Y|E\rangle \stackrel{c_1}{\rightarrow} \cdots \stackrel{c_n}{\rightarrow} \langle X|E\rangle$.

a or aX is an *exit* for cluster C if:

1. a or aX is a summand in the right-hand side of the recursive equation for a recursion variable in C, and
2. in the case of aX, $a \notin I \cup \{\tau\}$ or $X \notin C$.

CFAR If X is in a cluster for I with exits $\{v_1Y_1, \ldots, v_mY_m, w_1, \ldots, w_n\}$, then

$$v \cdot \tau_I(\langle X|E\rangle) =$$
$$v \cdot \tau_I(v_1\langle Y_1|E\rangle + \cdots + v_m\langle Y_m|E\rangle + w_1 + \cdots + w_n)$$

Intuition for CFAR: Let E denote

$$\begin{aligned} X_1 &= aX_2 + b_1 \\ &\vdots \\ X_{n-1} &= aX_n + b_{n-1} \\ X_n &= aX_1 + b_n \end{aligned}$$

$\tau_{\{a\}}(\langle X_1|E\rangle)$ executes τ-transitions until it executes an action b_i for $i \in \{1, \ldots, n\}$. After the execution of such a τ it is still possible to execute any of the b_i.

Fair abstraction says that $\tau_{\{a\}}(\langle X_1|E\rangle)$ does not stay in the τ-loop forever, so that at some time it will execute a b_i. Hence,

$$\tau_{\{a\}}(\langle X_1|E\rangle) \leftrightarrows_{rb} b_1 + \tau(b_1 + \cdots + b_n)$$

Namely, initially $\tau_{\{a\}}(\langle X_1|E\rangle)$ can execute b_1 or τ. In the latter case, this non-silent τ is followed by the execution of a series of silent τ's, until one of the actions b_i is executed.

Example

Let E denote

$$X \;=\; heads \cdot X + tails$$

$\langle X|E\rangle$ represents tossing a fair coin until the result is tails. We abstract away from throwing heads: $\tau_{\{heads\}}(\langle X|E\rangle)$.

$\{X\}$ is the only *cluster* for $\{heads\}$, and the only *exit* of this cluster is *tails*. So by CFAR,

$$\begin{aligned} \tau \cdot \tau_{\{heads\}}(\langle X|E\rangle) &= \tau \cdot \tau_{\{heads\}}(tails) \\ &= \tau \cdot tails \end{aligned}$$

Hence,

$$\begin{aligned} \tau_{\{heads\}}(\langle X|E\rangle) &= \tau_{\{heads\}}(heads \cdot \langle X|E\rangle + tails) \\ &= \tau \cdot \tau_{\{heads\}}(\langle X|E\rangle) + tails \\ &= \tau \cdot tails + tails \end{aligned}$$

Soundness: The axioms for ACP_τ with guarded linear recursion are *sound* modulo rooted branching bisimulation equivalence:

$$s = t \;\Rightarrow\; s \mathrel{\underline{\leftrightarrow}}_{rb} t$$

Rooted branching bisimulation equivalence is a congruence over ACP_τ with guarded linear recursion, so it suffices to check soundness of all instantiations of axioms.

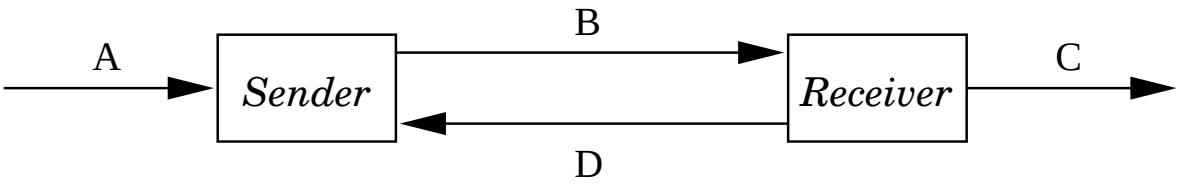
Completeness: The axioms for ACP_τ with guarded linear recursion are *complete* modulo rooted branching bisimulation equivalence:

$$s \mathrel{\underline{\leftrightarrow}}_{rb} t \;\Rightarrow\; s = t$$

This follows from two facts:

- Each term over ACP_τ with guarded linear recursion is equal to a term $\langle X|E\rangle$ with E a guarded linear recursive specification.
- If $\langle X|E\rangle \mathrel{\underline{\leftrightarrow}} \langle Y|E'\rangle$ for guarded linear recursive specifications E, E', then $\langle X|E\rangle = \langle Y|E'\rangle$.

Alternating Bit Protocol



Data elements are sent from *Sender* to *Receiver* via faulty channel B. *Sender* alternatingly attaches bit 0 or bit 1 to data elements.

If *Receiver* receives a datum, it sends the attached bit to *Sender* via faulty channel D, to acknowledge reception. If *Receiver* receives a corrupted message, then it resends the preceding acknowledgement.

Sender keeps sending a datum with attached bit b until it receives acknowledgement b. Then it starts sending the next datum with attached bit $1 - b$ until it receives acknowledgement $1 - b$, et cetera.

Sender sending a datum with bit b attached:

$$\begin{aligned} S_b &= \sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot T_{db} \\ T_{db} &= (s_{\mathrm{B}}(d,b)+s_{\mathrm{B}}(\bot))\cdot U_{db} \\ U_{db} &= r_{\mathrm{D}}(b)\cdot S_{1-b} \;+\; (r_{\mathrm{D}}(1-b)+r_{\mathrm{D}}(\bot))\cdot T_{db} \end{aligned}$$

Receiver expecting a datum with b attached:

$$\begin{aligned} R_b &= \sum_{d\in\Delta}\{r_{\mathrm{B}}(d,b)\cdot s_{\mathrm{C}}(d)\cdot Q_b \\ &\quad + r_{\mathrm{B}}(d,1-b)\cdot Q_{1-b}\} + r_{\mathrm{B}}(\bot)\cdot Q_{1-b} \\ Q_b &= (s_{\mathrm{D}}(b)+s_{\mathrm{D}}(\bot))\cdot R_{1-b} \end{aligned}$$

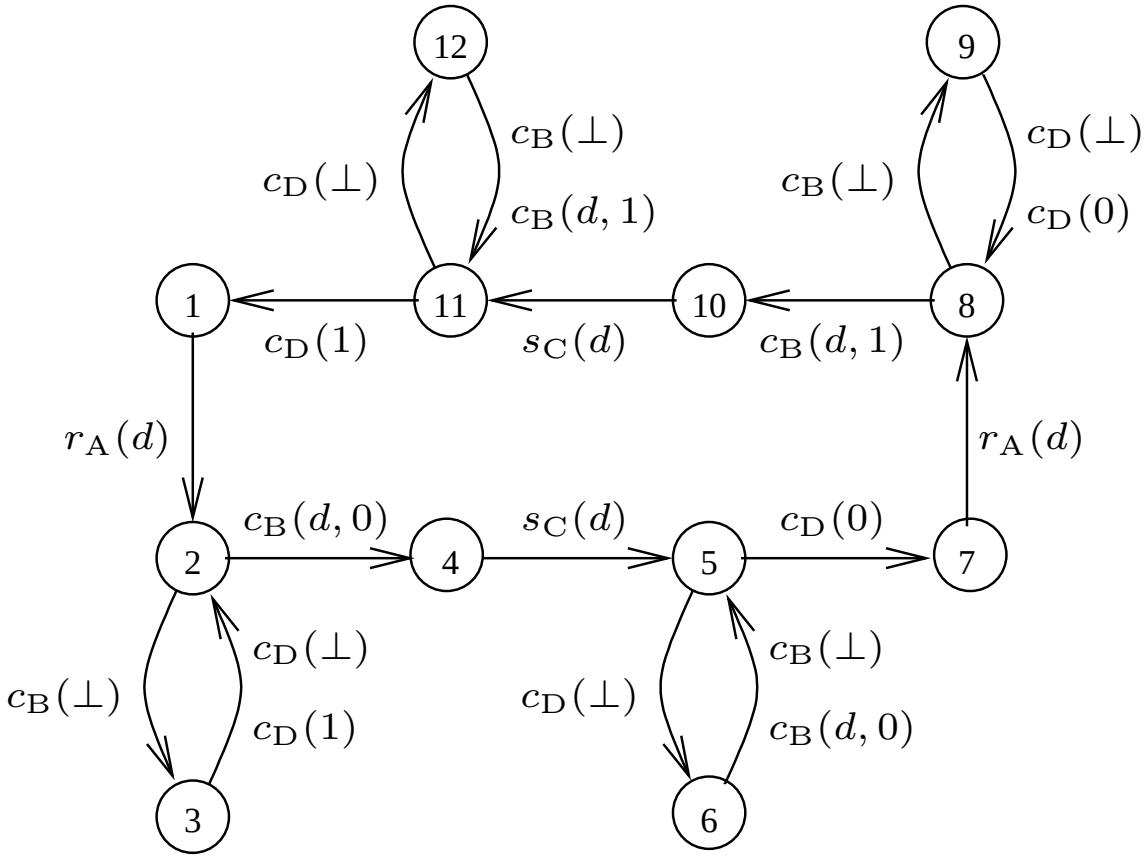
A send and a read action of the same message ((d,b), b or $\bot$) over the same channel (B or D) communicate with each other.

The alternating bit protocol is specified by

$$\tau_I(\partial_H(R_0\|S_0))$$

with H the set of read and send actions over channels B and D, and I the set of communication actions.

Transition graph of $\partial_H(R_0 \| S_0)$:



$$
\begin{aligned}
R_0 \| S_0 \;&=\; \textstyle\sum_{d' \in \Delta} \{ r_{\mathrm{B}}(d', 0) \cdot ((s_{\mathrm{C}}(d') Q_0) \| S_0) \\
&\quad + r_{\mathrm{B}}(d', 1) \cdot (Q_1 \| S_0) \} + r_{\mathrm{B}}(\bot) \cdot (Q_1 \| S_0) \\
&\quad + \textstyle\sum_{d \in \Delta} r_{\mathrm{A}}(d) \cdot (T_{d0} \| R_0) \\
\partial_H(R_0 \| S_0) \;&=\; \textstyle\sum_{d \in \Delta} r_{\mathrm{A}}(d) \cdot \partial_H(T_{d0} \| R_0)
\end{aligned}
$$

$$
\begin{aligned}
T_{d0} \| R_0 \;&=\; (s_{\mathrm{B}}(d, 0) + s_{\mathrm{B}}(\bot)) \cdot (U_{d0} \| R_0) \\
&\quad + \textstyle\sum_{d' \in \Delta} \{ r_{\mathrm{B}}(d', 0) \cdot ((s_{\mathrm{C}}(d') Q_0) \| T_{d0}) \\
&\quad + r_{\mathrm{B}}(d', 1) \cdot (Q_1 \| T_{d0}) \} + r_{\mathrm{B}}(\bot) \cdot (Q_1 \| T_{d0}) \\
&\quad + c_{\mathrm{B}}(d, 0) \cdot (U_{d0} \| (s_{\mathrm{C}}(d) Q_0)) + c_{\mathrm{B}}(\bot) \cdot (U_{d0} \| Q_1) \\
\partial_H(T_{d0} \| R_0) \;&=\; c_{\mathrm{B}}(d, 0) \cdot \partial_H(U_{d0} \| (s_{\mathrm{C}}(d) Q_0)) \\
&\qquad + c_{\mathrm{B}}(\bot) \cdot \partial_H(U_{d0} \| Q_1)
\end{aligned}
$$

$$
\begin{aligned}
U_{d0} \| Q_1 \;&=\; r_{\mathrm{D}}(0) \cdot (S_1 \| Q_1) \\
&\quad + (r_{\mathrm{D}}(1) + r_{\mathrm{D}}(\bot)) \cdot (T_{d0} \| Q_1) \\
&\quad + (s_{\mathrm{D}}(1) + s_{\mathrm{D}}(\bot)) \cdot (R_0 \| U_{d0}) \\
&\quad + (c_{\mathrm{D}}(1) + c_{\mathrm{D}}(\bot)) \cdot (T_{d0} \| R_0) \\
\partial_H(U_{d0} \| Q_1) \;&=\; (c_{\mathrm{D}}(1) + c_{\mathrm{D}}(\bot)) \cdot \partial_H(T_{d0} \| R_0)
\end{aligned}
$$

$$
\begin{aligned}
U_{d0} \| (s_{\mathrm{C}}(d) Q_0) \;&=\; r_{\mathrm{D}}(0) \cdot (S_1 \| (s_{\mathrm{C}}(d) Q_0)) \\
&\quad + (r_{\mathrm{D}}(1) + r_{\mathrm{D}}(\bot)) \cdot (T_{d0} \| (s_{\mathrm{C}}(d) Q_0)) \\
&\quad + s_{\mathrm{C}}(d) \cdot (Q_0 \| U_{d0}) \\
\partial_H (U_{d0} \| (s_{\mathrm{C}}(d) Q_0)) \;&=\; s_{\mathrm{C}}(d) \cdot \partial_H (Q_0 \| U_{d0})
\end{aligned}
$$

$$
\begin{aligned}
Q_0 \| U_{d0} \;&=\; (s_{\mathrm{D}}(0) + s_{\mathrm{D}}(\bot)) \cdot (R_1 \| U_{d0}) \\
&\quad + r_{\mathrm{D}}(0) \cdot (S_1 \| Q_0) + (r_{\mathrm{D}}(1) + r_{\mathrm{D}}(\bot)) \cdot (T_{d0} \| Q_0) \\
&\quad + c_{\mathrm{D}}(0) \cdot (R_1 \| S_1) + c_{\mathrm{D}}(\bot) \cdot (R_1 \| T_{d0}) \\
\partial_H (Q_0 \| U_{d0}) \;&=\; c_{\mathrm{D}}(0) \cdot \partial_H (R_1 \| S_1) \\
&\quad + c_{\mathrm{D}}(\bot) \cdot \partial_H (R_1 \| T_{d0})
\end{aligned}
$$

$$
\begin{aligned}
R_1 \| T_{d0} \;&=\; \sum_{d' \in \Delta} \{ r_{\mathrm{B}}(d', 1) \cdot ((s_{\mathrm{C}}(d') Q_1) \| T_{d0}) \\
&\quad + r_{\mathrm{B}}(d', 0) \cdot (Q_0 \| T_{d0}) \} + r_{\mathrm{B}}(\bot) \cdot (Q_0 \| T_{d0}) \\
&\quad + (s_{\mathrm{B}}(d, 0) + s_{\mathrm{B}}(\bot)) \cdot (U_{d0} \| R_1) \\
&\quad + (c_{\mathrm{B}}(d, 0) + c_{\mathrm{B}}(\bot)) \cdot (Q_0 \| U_{d0}) \\
\partial_H (R_1 \| T_{d0}) \;&=\; (c_{\mathrm{B}}(d, 0) + c_{\mathrm{B}}(\bot)) \cdot \partial_H (Q_0 \| U_{d0})
\end{aligned}
$$

Likewise we derive

$$\partial_H(R_1\|S_1) \;=\; \textstyle\sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot\partial_H(T_{d1}\|R_1)$$

$$\begin{aligned}\partial_H(T_{d1}\|R_1) \;&=\; c_{\mathrm{B}}(d,1)\cdot\partial_H(U_{d1}\|(s_{\mathrm{C}}(d)\cdot Q_1))\\ &\quad+\; c_{\mathrm{B}}(\perp)\cdot\partial_H(U_{d1}\|Q_0)\end{aligned}$$

$$\partial_H(U_{d1}\|Q_0) \;=\; (c_{\mathrm{D}}(0)+c_{\mathrm{D}}(\perp))\cdot\partial_H(T_{d1}\|R_1)$$

$$\partial_H(U_{d1}\|(s_{\mathrm{C}}(d)Q_1)) \;=\; s_{\mathrm{C}}(d)\cdot\partial_H(Q_1\|U_{d1})$$

$$\begin{aligned}\partial_H(Q_1\|U_{d1}) \;&=\; c_{\mathrm{D}}(1)\cdot\partial_H(R_0\|S_0)\\ &\quad+\; c_{\mathrm{D}}(\perp)\cdot\partial_H(R_0\|T_{d1})\end{aligned}$$

$$\partial_H(R_0\|T_{d1}) \;=\; (c_{\mathrm{B}}(d,1)+c_{\mathrm{B}}(\perp))\cdot\partial_H(Q_1\|U_{d1})$$

Owing to the twelve derived equations, by RSP

$$\partial_H(R_0 \| S_0) \;=\; \langle X_1 | E \rangle$$

where E is the linear recursive specification

$$\begin{array}{lll}
\{ \quad X_1 & = & \sum_{d' \in \Delta} r_{\mathrm{A}}(d') \cdot X_{2d'} \\
\quad X_{2d} & = & c_{\mathrm{B}}(d,0) \cdot X_{4d} + c_{\mathrm{B}}(\bot) \cdot X_{3d} \\
\quad X_{3d} & = & (c_{\mathrm{D}}(1) + c_{\mathrm{D}}(\bot)) \cdot X_{2d} \\
\quad X_{4d} & = & s_{\mathrm{C}}(d) \cdot X_{5d} \\
\quad X_{5d} & = & c_{\mathrm{D}}(0) \cdot Y_1 + c_{\mathrm{D}}(\bot) \cdot X_{6d} \\
\quad X_{6d} & = & (c_{\mathrm{B}}(d,0) + c_{\mathrm{B}}(\bot)) \cdot X_{5d} \\
\quad Y_1 & = & \sum_{d' \in \Delta} r_{\mathrm{A}}(d') \cdot Y_{2d'} \\
\quad Y_{2d} & = & c_{\mathrm{B}}(d,1) \cdot Y_{4d} + c_{\mathrm{B}}(\bot) \cdot Y_{3d} \\
\quad Y_{3d} & = & (c_{\mathrm{D}}(0) + c_{\mathrm{D}}(\bot)) \cdot Y_{2d} \\
\quad Y_{4d} & = & s_{\mathrm{C}}(d) \cdot Y_{5d} \\
\quad Y_{5d} & = & c_{\mathrm{D}}(1) \cdot X_1 + c_{\mathrm{D}}(\bot) \cdot Y_{6d} \\
\quad Y_{6d} & = & (c_{\mathrm{B}}(d,1) + c_{\mathrm{B}}(\bot)) \cdot Y_{5d} \\
\quad | \quad d \in \Delta & & \}
\end{array}$$

After application of τ_I to $\langle X_1|E\rangle$, the loops of communication actions become τ-loops, which can be removed by CFAR.

For example, X_{2d} and X_{3d} form a cluster for I with exit $c_{\mathrm{B}}(d,0)\cdot X_{4d}$, so

$$\begin{aligned} & r_{\mathrm{A}}(d)\cdot\tau_I(\langle X_{2d}|E\rangle) \\ = \;& r_{\mathrm{A}}(d)\cdot\tau_I(c_{\mathrm{B}}(d,0)\cdot\langle X_{4d}|E\rangle) \\ = \;& r_{\mathrm{A}}(d)\cdot\tau\cdot\tau_I(\langle X_{4d}|E\rangle) \\ = \;& r_{\mathrm{A}}(d)\cdot\tau_I(\langle X_{4d}|E\rangle) \end{aligned}$$

Likewise we derive

$$\begin{aligned} s_{\mathrm{C}}(d)\cdot\tau_I(\langle X_{5d}|E\rangle) &= s_{\mathrm{C}}(d)\cdot\tau_I(\langle Y_1|E\rangle) \\ r_{\mathrm{A}}(d)\cdot\tau_I(\langle Y_{2d}|E\rangle) &= r_{\mathrm{A}}(d)\cdot\tau_I(\langle Y_{4d}|E\rangle) \\ s_{\mathrm{C}}(d)\cdot\tau_I(\langle Y_{5d}|E\rangle) &= s_{\mathrm{C}}(d)\cdot\tau_I(\langle X_1|E\rangle) \end{aligned}$$

$$\begin{aligned}
\tau_I(\langle X_1|E\rangle) &= \textstyle\sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot\tau_I(\langle X_{2d}|E\rangle)\\
&= \textstyle\sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot\tau_I(\langle X_{4d}|E\rangle)\\
&= \textstyle\sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot s_{\mathrm{C}}(d)\cdot\tau_I(\langle X_{5d}|E\rangle)\\
&= \textstyle\sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot s_{\mathrm{C}}(d)\cdot\tau_I(\langle Y_1|E\rangle)
\end{aligned}$$

Likewise we derive

$$\tau_I(\langle Y_1|E\rangle) = \textstyle\sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot s_{\mathrm{C}}(d)\cdot\tau_I(\langle X_1|E\rangle)$$

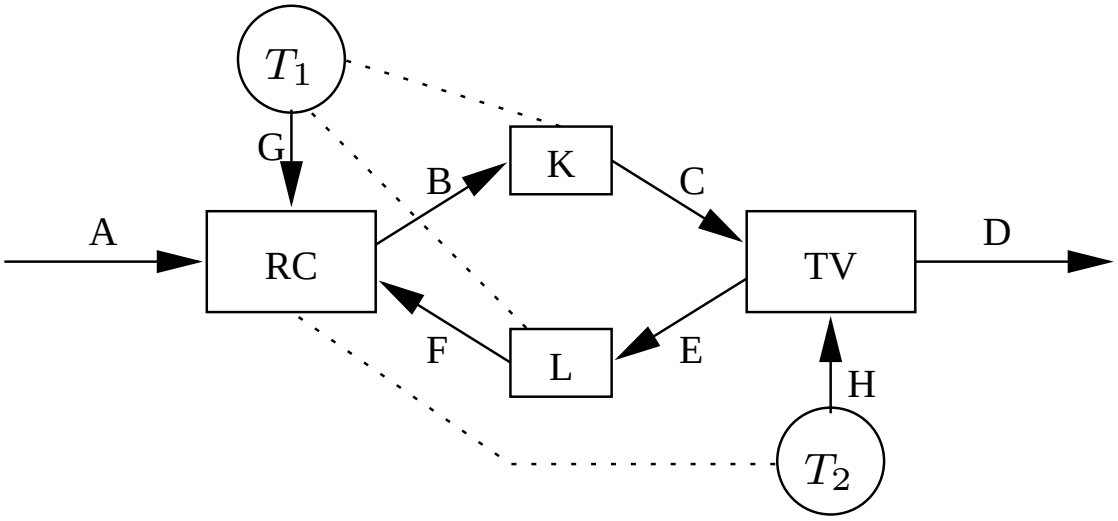
So using RSP it follows that

$$\tau_I(\langle X_1|E\rangle) = \langle Z \mid Z = r_{\mathrm{A}}(d)\cdot s_{\mathrm{C}}(d)\cdot Z\rangle$$

Hence,

$$\begin{aligned}
&\tau_I(\partial_H(R_0\|S_0))\\
=\ &\textstyle\sum_{d\in\Delta} r_{\mathrm{A}}(d)\cdot s_{\mathrm{C}}(d)\cdot\tau_I(\partial_H(R_0\|S_0))
\end{aligned}$$

Bounded Retransmission Protocol



Data *packets* are sent from RC to TV.

A datum may get *lost*.

An alternating bit is attached to each datum.

Only *one* kind of acknowledgement.

A datum is resent a *limited* number of times.

A label is attached to *last* datum of the packet.

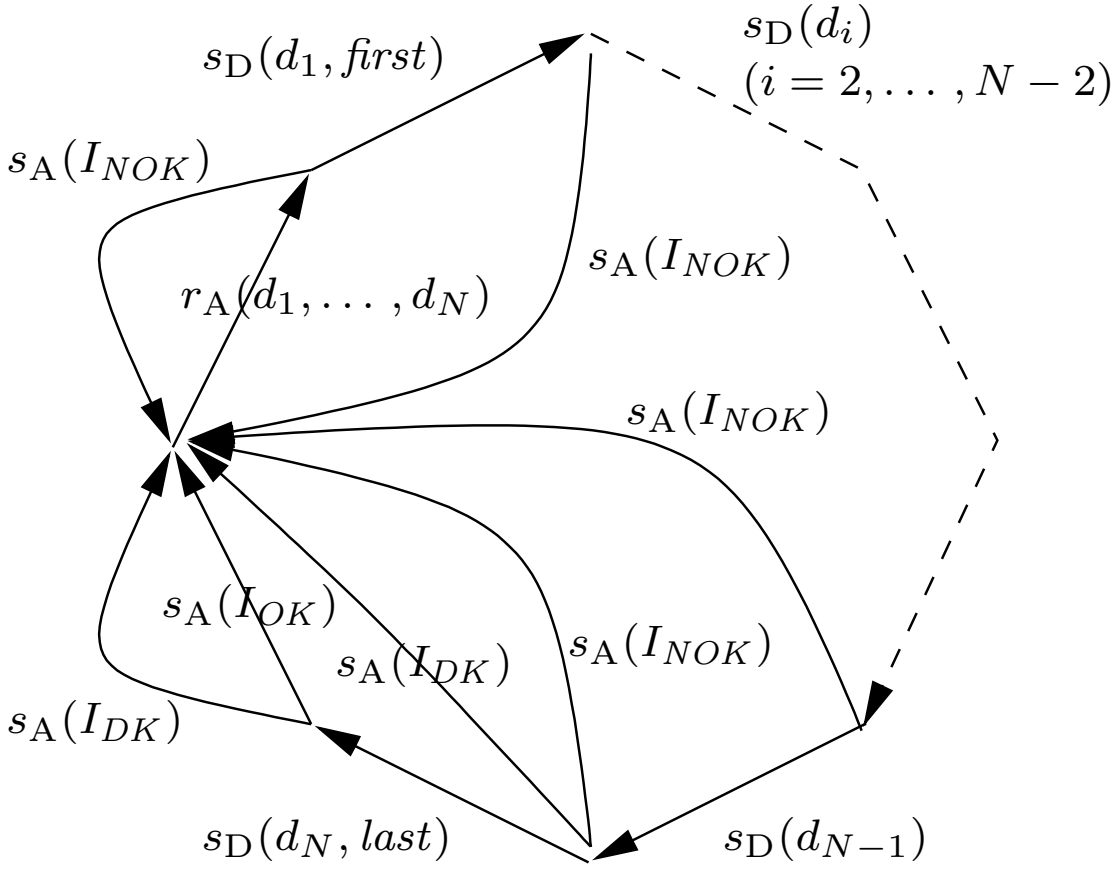
If RC does not receive an acknowledgement within a certain *time*, it resends the datum.

If TV does not receive a next datum within a certain *time*, RC has given up transmission.

T_1 and T_2 send *time-out* messages.

K and L model *non-determinism* of medium.

External Behaviour



Error Messages

$s_{\mathrm{A}}(I_{NOK})$: RC warns that transmission was unsuccessful

$s_{\mathrm{A}}(I_{DK})$: RC warns that transmission *may* have been unsuccessful

Remote Control

$$
\begin{aligned}
X &= \textstyle\sum_{\ell \in \Lambda} r_{\mathrm{A}}(\ell) \cdot Y(\ell, 0, 0) \\
\\
Y(\ell, b, n) &= s_{\mathrm{B}}(head(\ell), b) \cdot Z(\ell, b, n) \\
Y(d, b, n) &= s_{\mathrm{B}}(d, b, last) \cdot Z(d, b, n) \\
\\
n < max &: \\
Z(\ell, b, n) &= r_{\mathrm{F}}(ack) \cdot Y(tail(\ell), 1 - b, 0) \\
&+ r_{\mathrm{G}}(to) \cdot Y(\ell, b, n + 1) \\
Z(\ell, b, max) &= r_{\mathrm{F}}(ack) \cdot Y(tail(\ell), 1 - b, 0) \\
&+ r_{\mathrm{G}}(to) \cdot s_{\mathrm{A}}(I_{NOK}) \cdot s_{\mathrm{H}}(to) \cdot X \\
\\
n < max &: \\
Z(d, b, n) &= r_{\mathrm{F}}(ack) \cdot s_{\mathrm{A}}(I_{OK}) \cdot X \\
&+ r_{\mathrm{G}}(to) \cdot Y(d, b, n + 1) \\
Z(d, b, max) &= r_{\mathrm{F}}(ack) \cdot s_{\mathrm{A}}(I_{OK}) \cdot X \\
&+ r_{\mathrm{G}}(to) \cdot s_{\mathrm{A}}(I_{DK}) \cdot s_{\mathrm{H}}(to) \cdot X
\end{aligned}
$$

Television

$$
\begin{aligned}
V \;=\; & \textstyle\sum_{d\in\Delta} r_{\mathrm{C}}(d,0)\cdot s_{\mathrm{D}}(d,\mathit{first})\cdot s_{\mathrm{E}}(\mathit{ack})\cdot W(1)\\
+\; & \textstyle\sum_{d\in\Delta}(r_{\mathrm{C}}(d,0,\mathit{last})+r_{\mathrm{C}}(d,1,\mathit{last}))\cdot s_{\mathrm{E}}(\mathit{ack})\cdot V\\
+\; & r_{\mathrm{H}}(\mathit{to})\cdot V
\end{aligned}
$$

$$
\begin{aligned}
W(b) \;=\; & \textstyle\sum_{d\in\Delta} r_{\mathrm{C}}(d,b)\cdot s_{\mathrm{D}}(d)\cdot s_{\mathrm{E}}(\mathit{ack})\cdot W(1-b)\\
& +\; \textstyle\sum_{d\in\Delta} r_{\mathrm{C}}(d,b,\mathit{last})\cdot s_{\mathrm{D}}(d,\mathit{last})\cdot s_{\mathrm{E}}(\mathit{ack})\cdot V\\
& +\; \textstyle\sum_{d\in\Delta} r_{\mathrm{C}}(d,1-b)\cdot s_{\mathrm{E}}(\mathit{ack})\cdot W(b)\\
& +\; r_{\mathrm{H}}(\mathit{to})\cdot V
\end{aligned}
$$

Medium

$$
\begin{aligned}
K \;=\; & \textstyle\sum_{d\in\Delta}\sum_{b\in\{0,1\}}\\
& \{r_{\mathrm{B}}(d,b)\cdot(s_{\mathrm{C}}(d,b)+s_{\mathrm{G}}(\mathit{to}))\cdot K\\
& +\; r_{\mathrm{B}}(d,b,\mathit{last})\cdot(s_{\mathrm{C}}(d,b,\mathit{last})+s_{\mathrm{G}}(\mathit{to}))\cdot K\}\\
L \;=\; & r_{\mathrm{E}}(\mathit{ack})\cdot(s_{\mathrm{F}}(\mathit{ack})+s_{\mathrm{G}}(\mathit{to}))\cdot L
\end{aligned}
$$

Bounded Retransmission Protocol

$$\tau_I(\partial_H(V\|X\|K\|L))$$

with H the internal read and send actions, and I the communication actions.

Initial state (length of ℓ at least two):

$$\partial_H(V\|X\|K\|L) \;=\; \textstyle\sum_{\ell\in\Lambda} r_{\mathrm{A}}(\ell)\cdot \\ \partial_H(V\|Y(\ell,0,0)\|K\|L)$$

RC sends the first datum of a packet, TV has not yet received this datum:

$$\partial_H(V\|Y(\ell,0,n)\|K\|L) \;=\; c_{\mathrm{B}}(head(\ell),0)\cdot \\ \partial_H(V\|Z(\ell,0,n)\|K'(head(\ell),0)\|L)$$

RC sends a non-last datum of a packet, TV already received some data of this packet:

$$\partial_H(W(b)\|Y(\ell,b',n)\|K\|L) \;=\; c_{\mathrm{B}}(head(\ell),b')\cdot \\ \partial_H(W(b)\|Z(\ell,b',n)\|K'(head(\ell),b')\|L)$$

RC sends the last datum of a packet. Two cases depend on whether TV received this datum:

$$\partial_H(W(b)\|Y(d,b,n)\|K\|L) \;=\; c_{\mathrm{B}}(d,b,last)\cdot \\ \partial_H(W(b)\|Z(d,b,n)\|K'(d,b,last)\|L)$$

$$\partial_H(V\|Y(d,b,n)\|K\|L) \;=\; c_{\mathrm{B}}(d,b,last)\cdot \\ \partial_H(V\|Z(d,b,n)\|K'(d,b,last)\|L)$$

K may lose the first datum of a packet, TV did not yet receive this datum. Two cases depend on whether the counter reached its maximum value:

$$
\begin{aligned}
&\partial_H(V\|Z(\ell,0,n)\|K'(head(\ell),0)\|L) \;=\\
&\quad c_{\mathrm{C}}(head(\ell),0)\cdot s_{\mathrm{D}}(head(\ell),\mathit{first})\cdot c_{\mathrm{E}}(ack)\cdot\\
&\quad \partial_H(W(1)\|Z(\ell,0,n)\|K\|L')\\
&\quad + c_{\mathrm{G}}(to)\cdot\partial_H(V\|Y(\ell,0,n+1)\|K\|L)\\
&\partial_H(V\|Z(\ell,0,max)\|K'(head(\ell),0)\|L) \;=\\
&\quad c_{\mathrm{C}}(head(\ell),0)\cdot s_{\mathrm{D}}(head(\ell),\mathit{first})\cdot c_{\mathrm{E}}(ack)\cdot\\
&\quad \partial_H(W(1)\|Z(\ell,0,max)\|K\|L')\\
&\quad + c_{\mathrm{G}}(to)\cdot s_{\mathrm{A}}(I_{NOK})\cdot c_{\mathrm{H}}(to)\cdot\partial_H(V\|X\|K\|L)
\end{aligned}
$$

K may lose a datum of a packet, TV already received some data of the packet. Eight cases depend on whether the counter reached its maximum value, whether K handles the last datum of the packet, and whether TV already received the datum handled by K:

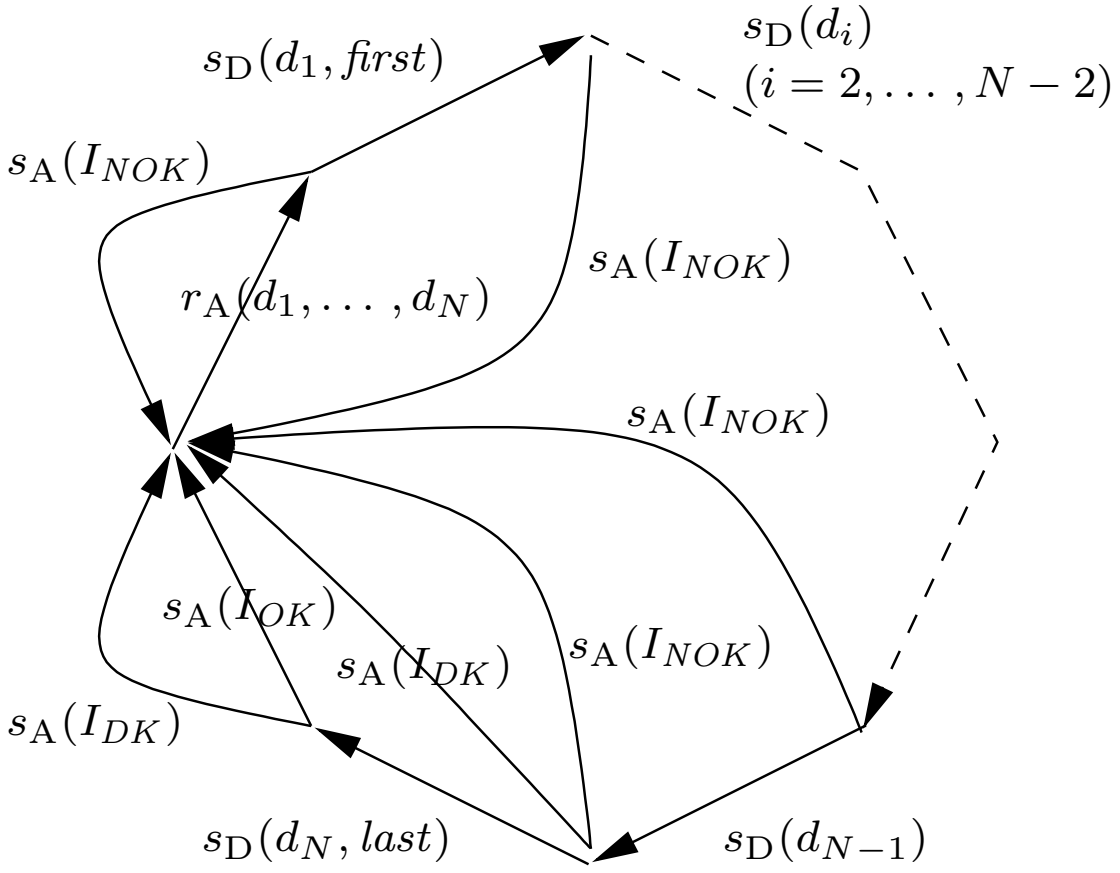
$$
\begin{aligned}
&\partial_H(W(b)\|Z(\ell,b,n)\|K'(head(\ell),b)\|L) \;=\\
&\qquad c_{\mathrm{C}}(head(\ell),b)\cdot s_{\mathrm{D}}(head(\ell))\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(W(1-b)\|Z(\ell,b,n)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot\partial_H(W(b)\|Y(\ell,b,n+1)\|K\|L)\\
&\partial_H(W(b)\|Z(\ell,b,max)\|K'(head(\ell),b)\|L) \;=\\
&\qquad c_{\mathrm{C}}(head(\ell),b)\cdot s_{\mathrm{D}}(head(\ell))\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(W(1-b)\|Z(\ell,b,max)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot s_{\mathrm{A}}(I_{NOK})\cdot c_{\mathrm{H}}(to)\;\cdot\\
&\qquad \partial_H(V\|X\|K\|L)\\
&\partial_H(W(b)\|Z(d,b,n)\|K'(d,b,last)\|L) \;=\\
&\qquad c_{\mathrm{C}}(d,b,last)\cdot s_{\mathrm{D}}(d,last)\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(V\|Z(d,b,n)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot\partial_H(W(b)\|Y(\ell,b,n+1)\|K\|L)\\
&\partial_H(W(b)\|Z(d,b,max)\|K'(d,b,last)\|L) \;=\\
&\qquad c_{\mathrm{C}}(d,b,last)\cdot s_{\mathrm{D}}(d,last)\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(V\|Z(d,b,max)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot s_{\mathrm{A}}(I_{DK})\cdot c_{\mathrm{H}}(to)\;\cdot\\
&\qquad \partial_H(V\|X\|K\|L)
\end{aligned}
$$

$$
\begin{aligned}
&\partial_H(W(b)\|Z(\ell,1-b,n)\|K'(head(\ell),1-b)\|L) \;=\\
&\qquad c_{\mathrm{C}}(head(\ell),1-b)\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(W(b)\|Z(\ell,1-b,n)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot\partial_H(W(b)\|Y(\ell,1-b,n+1)\|K\|L)\\
&\partial_H(W(b)\|Z(\ell,1-b,max)\|K'(head(\ell),1-b)\|L) =\\
&\qquad c_{\mathrm{C}}(head(\ell),1-b)\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(W(b)\|Z(\ell,1-b,max)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot s_{\mathrm{A}}(I_{NOK})\cdot c_{\mathrm{H}}(to)\;\cdot\\
&\qquad \partial_H(V\|X\|K\|L)\\
&\partial_H(V\|Z(d,b,n)\|K'(d,b,last)\|L) \;=\\
&\qquad c_{\mathrm{C}}(d,b,last)\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(V\|Z(d,b,n)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot\partial_H(V\|Y(d,b,n+1)\|K\|L)\\
&\partial_H(V\|Z(d,b,max)\|K'(d,b,last)\|L) \;=\\
&\qquad c_{\mathrm{C}}(d,b,last)\cdot c_{\mathrm{E}}(ack)\;\cdot\\
&\qquad \partial_H(V\|Z(d,b,max)\|K\|L')\\
&\qquad +\; c_{\mathrm{G}}(to)\cdot s_{\mathrm{A}}(I_{DK})\cdot c_{\mathrm{H}}(to)\cdot\partial_H(V\|X\|K\|L)
\end{aligned}
$$

L may lose a datum of a packet. Four cases depend on whether the counter reached its maximum value and whether K handles the last datum of the packet:

$$
\begin{aligned}
&\partial_H(W(b)\|Z(\ell,1-b,n)\|K\|L') \;=\\
&\qquad c_{\mathrm{F}}(ack)\cdot\partial_H(W(b)\|Y(tail(\ell),b,0)\|K\|L)\\
&\qquad +\, c_{\mathrm{G}}(to)\cdot\partial_H(W(b)\|Y(\ell,1-b,n+1)\|K\|L)\\
&\partial_H(W(b)\|Z(\ell,1-b,max)\|K\|L') \;=\\
&\qquad c_{\mathrm{F}}(ack)\cdot\partial_H(W(b)\|Y(tail(\ell),b,0)\|K\|L)\\
&\qquad +\, c_{\mathrm{G}}(to)\cdot s_{\mathrm{A}}(I_{NOK})\cdot c_{\mathrm{H}}(to)\cdot\\
&\qquad \partial_H(V\|X\|K\|L)\\
&\partial_H(V\|Z(d,b,n)\|K\|L') \;=\\
&\qquad c_{\mathrm{F}}(ack)\cdot s_{\mathrm{A}}(I_{OK})\cdot\partial_H(V\|X\|K\|L)\\
&\qquad +\, c_{\mathrm{G}}(to)\cdot\partial_H(V\|Y(d,b,n+1)\|K\|L)\\
&\partial_H(V\|Z(d,b,max)\|K\|L') \;=\\
&\qquad c_{\mathrm{F}}(ack)\cdot s_{\mathrm{A}}(I_{OK})\cdot\partial_H(V\|X\|K\|L)\\
&\qquad +\, c_{\mathrm{G}}(to)\cdot s_{\mathrm{A}}(I_{DK})\cdot c_{\mathrm{H}}(to)\;\cdot\partial_H(V\|X\|K\|L)
\end{aligned}
$$

After application of the abstraction operator τ_I, communication actions over the internal channels are renamed into τ. Some of these τ's are non-silent, so that we obtain the desired external behaviour



intertwined with τ-transitions.

Renaming

The *renaming* operator ρ_f, with $f : A \to A$, renames all actions $a \in A$ in its argument into $f(a)$.

$$\frac{x \xrightarrow{v} \surd}{\rho_f(x) \xrightarrow{f(v)} \surd} \qquad \frac{x \xrightarrow{v} x'}{\rho_f(x) \xrightarrow{f(v)} \rho_f(x')}$$

with $f(\tau) = \tau$.

Conservative Extension: ACP_τ with guarded linear recursion and renaming is a *conservative extension* of ACP_τ with guarded linear recursion.

Congruence: Rooted branching bisimulation equivalence is a *congruence* over ACP_τ with guarded linear recursion and renaming.

RN1	$\rho_f(v)$	$=$	$f(v)$
RN2	$\rho_f(\delta)$	$=$	δ
RN3	$\rho_f(x+y)$	$=$	$\rho_f(x)+\rho_f(y)$
RN4	$\rho_f(x\cdot y)$	$=$	$\rho_f(x)\cdot\rho_f(y)$

Soundness: The axioms for ACP_τ with guarded linear recursion and renaming are *sound* modulo rooted branching bisimulation equivalence:

$$s = t \;\Rightarrow\; s \mathrel{\underline{\leftrightarrow}}_{rb} t$$

Completeness: The axioms for ACP_τ with guarded linear recursion and renaming are *complete* modulo rooted branching bisimulation equivalence:

$$s \mathrel{\underline{\leftrightarrow}}_{rb} t \;\Rightarrow\; s = t$$

State Operator

Let S be a set of states.

$$\begin{aligned} \mathit{action}: &\quad S \times A \quad \to \quad A \\ \mathit{effect}: &\quad S \times A \quad \to \quad S \end{aligned}$$

The *state operator* $\lambda_s(t)$ denotes term t in state s:

$$\frac{x \xrightarrow{v} \surd}{\lambda_s(x) \overset{action(s,v)}{\to} \surd} \qquad \frac{x \xrightarrow{v} x'}{\lambda_s(x) \overset{action(s,v)}{\to} \lambda_{effect(s,v)}(x')}$$

with $action(s,\tau) = \tau$ and $effect(s,\tau) = s$.

Conservative Extension: ACP_τ with guarded linear recursion and state operator is a *conservative extension* of ACP_τ with guarded linear recursion.

Congruence: Rooted branching bisimulation equivalence is a *congruence* over ACP_τ with guarded linear recursion and state operator.

$$
\begin{array}{lrcl}
\text{SO1} & \lambda_s(v) & = & action(s,v) \\
\text{SO2} & \lambda_s(\delta) & = & \delta \\
\text{SO3} & \lambda_s(x+y) & = & \lambda_s(x)+\lambda_s(y) \\
\text{SO4} & \lambda_s(v\cdot y) & = & action(s,v)\cdot\lambda_{effect(s,v)}(y)
\end{array}
$$

Soundness: The axioms for ACP_τ with guarded linear recursion and state operator are *sound* modulo rooted branching bisimulation equivalence:

$$s = t \;\Rightarrow\; s \mathrel{\underline{\leftrightarrow}_{rb}} t$$

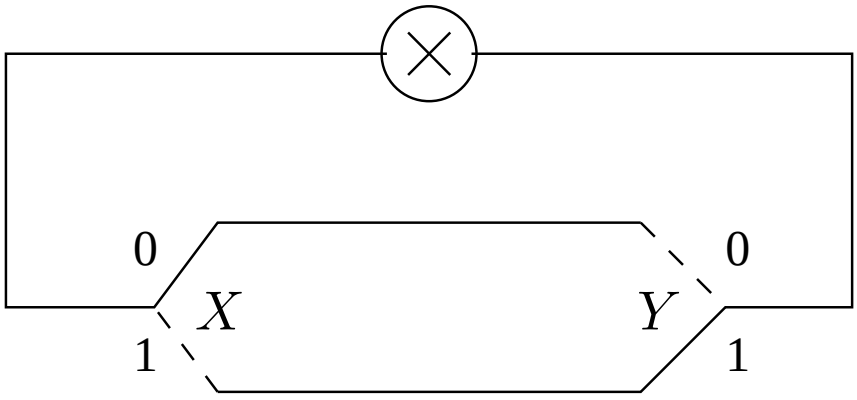
Completeness: The axioms for ACP_τ with guarded linear recursion and state operator are *complete* modulo rooted branching bisimulation equivalence:

$$s \mathrel{\underline{\leftrightarrow}_{rb}} t \;\Rightarrow\; s = t$$

Example: A light that can be switched on and off at two locations X and Y, which can be in two positions 0 and 1. The set of states is

$$\{\langle i, j\rangle \mid i, j \in \{0, 1\}\}$$

In $\langle i, j\rangle$ switch X is in position i and switch Y is in position j. The light is on if X and Y are in the same position 0 or 1. Initially, X is in position 0 and Y is in position 1:



Actions a and b represent flipping the switches at locations X and Y, respectively.

$$\begin{aligned} X &= aX \\ Y &= bY \end{aligned}$$

Actions *on* and *off* represent turning the light on and off.

$$
\begin{array}{lcl}
action(\langle i, i\rangle, a) & \equiv & off \\
action(\langle i, 1-i\rangle, a) & \equiv & on \\
action(\langle i, i\rangle, b) & \equiv & off \\
action(\langle i, 1-i\rangle, b) & \equiv & on \\
effect(\langle i, j\rangle, a) & \equiv & \langle 1-i, j\rangle \\
effect(\langle i, j\rangle, b) & \equiv & \langle i, 1-j\rangle
\end{array}
$$

The initial situation is captured by $\lambda_{\langle 0,1\rangle}(t)$, where t denotes $\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle$.

$$
\begin{array}{lcl}
\lambda_{\langle 0,1\rangle}(t) & = & \lambda_{\langle 0,1\rangle}(at + bt) \\
& = & \lambda_{\langle 0,1\rangle}(at) + \lambda_{\langle 0,1\rangle}(bt) \\
& = & action(\langle 0,1\rangle, a) \cdot \lambda_{effect(\langle 0,1\rangle, a)}(t) \\
& + & action(\langle 0,1\rangle, b) \cdot \lambda_{effect(\langle 0,1\rangle, b)}(t) \\
& \equiv & on \cdot \lambda_{\langle 1,1\rangle}(t) + on \cdot \lambda_{\langle 0,0\rangle}(t)
\end{array}
$$

Likewise we derive

$$
\begin{array}{lcllll}
\lambda_{\langle 1,0\rangle}(t) & = & on \cdot \lambda_{\langle 1,1\rangle}(t) & + & on \cdot \lambda_{\langle 0,0\rangle}(t) \\
\lambda_{\langle 1,1\rangle}(t) & = & off \cdot \lambda_{\langle 0,1\rangle}(t) & + & off \cdot \lambda_{\langle 1,0\rangle}(t) \\
\lambda_{\langle 0,0\rangle}(t) & = & off \cdot \lambda_{\langle 0,1\rangle}(t) & + & off \cdot \lambda_{\langle 1,0\rangle}(t)
\end{array}
$$

Let E denote

$$\begin{aligned} Z_1 &= on \cdot Z_3 + on \cdot Z_4 \\ Z_2 &= on \cdot Z_3 + on \cdot Z_4 \\ Z_3 &= off \cdot Z_1 + off \cdot Z_2 \\ Z_4 &= off \cdot Z_1 + off \cdot Z_2 \end{aligned}$$

By RSP,

$$\lambda_{\langle 0,1 \rangle}(t) \;=\; \langle Z_1 | E \rangle$$

Furthermore, by RSP,

$$\langle W \,|\, W{=}on{\cdot}off{\cdot}W \rangle \;=\; \langle Z_1 | E \rangle$$

Hence,

$$\lambda_{\langle 0,1 \rangle}(t) \;=\; \langle W \,|\, W{=}on{\cdot}off{\cdot}W \rangle$$

So by RDP,

$$\lambda_{\langle 0,1 \rangle}(t) \;=\; on \cdot off \cdot \lambda_{\langle 0,1 \rangle}(t)$$

Priorities

Let $<$ be a partial order on $A \cup \{\tau\}$. $\Theta(t)$ executes all transitions of t for which there is no simultaneous transition of higher priority:

$$\frac{x \xrightarrow{v} \surd \quad x \overset{w}{\not\to} \text{ for } v < w}{\Theta(x) \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x' \quad x \overset{w}{\not\to} \text{ for } v < w}{\Theta(x) \xrightarrow{v} \Theta(x')}$$

An auxiliary *unless* operator $\triangleleft$ is needed to axiomatise priorities. $s \triangleleft t$ does not execute *initial* transitions of s for which there are simultaneous transitions of higher priority:

$$\frac{x \xrightarrow{v} \surd \quad y \overset{w}{\not\to} \text{ for } v < w}{x \triangleleft y \xrightarrow{v} \surd} \qquad \frac{x \xrightarrow{v} x' \quad y \overset{w}{\not\to} \text{ for } v < w}{x \triangleleft y \xrightarrow{v} x'}$$

Conservative Extension: ACP_τ with guarded linear recursion, priority and unless is a *conservative extension* of ACP_τ with guarded linear recursion.

Congruence: If τ has priority over any other action, then rooted branching bisimulation equivalence is a *congruence* over ACP_τ with guarded linear recursion, priority and unless.

If τ does not have priority over any other action, then rooted branching bisimulation may not be a congruence over ACP_τ with guarded linear recursion, priority and unless.

Example

Let A consist of $\{a, b, c\}$, and let the partial order on $A \cup \{\tau\}$ be $\{b < c\}$.

$$a(\tau(b + c) + b) \underline{\leftrightarrow}_{rb} a(b + c)$$

because the τ in the process term at the left-hand side is silent. However,

$$\begin{aligned} \Theta(a(\tau(b + c) + b)) &\underline{\leftrightarrow}_{rb} a(\tau c + b) \\ \Theta(a(b + c)) &\underline{\leftrightarrow}_{rb} ac \end{aligned}$$

and $a(\tau c + b)$ and ac are not rooted branching bisimilar.

$$
\begin{array}{lrcll}
\text{TH1} & \Theta(v) & = & v & \\
\text{TH2} & \Theta(\delta) & = & \delta & \\
\text{TH3} & \Theta(x+y) & = & \Theta(x) \lhd y + \Theta(y) \lhd x & \\
\text{TH4} & \Theta(x \cdot y) & = & \Theta(x) \cdot \Theta(y) &
\end{array}
$$

$$
\begin{array}{lrcll}
\text{P1} & v \lhd w & = & v & (v \not< w) \\
\text{P2} & v \lhd w & = & \delta & (v < w) \\
\text{P3} & v \lhd \delta & = & v & \\
\text{P4} & \delta \lhd v & = & \delta & \\
\text{P5} & (x+y) \lhd z & = & (x \lhd z) + (y \lhd z) & \\
\text{P6} & (x \cdot y) \lhd z & = & (x \lhd z) \cdot y & \\
\text{P7} & x \lhd (y+z) & = & (x \lhd y) \lhd z & \\
\text{P8} & x \lhd (y \cdot z) & = & x \lhd y &
\end{array}
$$

Soundness and Completeness: The axioms for ACP_τ with guarded linear recursion, priorities and unless are *sound* and *complete* modulo rooted branching bisimulation:

$$s = t \;\Leftrightarrow\; s \underline{\leftrightarrow}_{rb} t$$

Example: Let $\gamma(a,b) \equiv c$, $a < c$ and $b < c$.

$$
\begin{aligned}
& \partial_{\{a,b\}}(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \\
= \quad & \partial_{\{a,b\}}(a \cdot \langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle \\
& + b \cdot \langle Y \mid Y{=}bY\rangle \| \langle X \mid X{=}aX\rangle \\
& + c \cdot \langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \\
= \quad & c \cdot \partial_{\{a,b\}}(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle)
\end{aligned}
$$

$$
\begin{aligned}
& \theta(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \\
= \quad & \theta(a\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle \\
& + b\langle Y \mid Y{=}bY\rangle \| \langle X \mid X{=}aX\rangle \\
& + c\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \\
= \quad & c \cdot \theta(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle)
\end{aligned}
$$

So by RSP,

$$
\begin{aligned}
& \partial_{\{a,b\}}(\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle) \\
= \quad & \langle Z \mid Z{=}cZ\rangle \\
= \quad & \theta(a\langle X \mid X{=}aX\rangle \| \langle Y \mid Y{=}bY\rangle)
\end{aligned}
$$

http://www.springer.com/978-3-540-66579-3

Introduction to Process Algebra
Fokkink, W.
2000, VIII, 168 p., Hardcover
ISBN: 978-3-540-66579-3