Internship Report: Expressiveness of Distributed Systems

Anran Wang

1 Introduction

This is a report as a result of an internship under supervision of Dr. Rob van Glabbeek and Dr. Peter Höfner from 6th August 2018 to 12th October 2018.

This report can be roughly divided into two parts: the first part is focused on *concurrency*, where the semantics for ABC(Algebra of Broadcasting Communication)[1] with respect to Component-Labelled Transition System(CLTS) is given, with the purpose to distinguish just and unjust paths. The second part is focused on *expressiveness*, where an encoding of ABC to CCS with Priority(CCS_P)[1] is proposed, and proved adequate with respect to the criterion proposed by [5].

This report could be completed by some follow-up works like: Revise the CLTS semantics for ABC to have more expressiveness power in the sense that it can distinguish more paths that are unjust; reconsider the CLTS semantics for CCS_P; complete the proof for the adequacy of the encoding from ABC to CCS_P; find and prove separation results to prove that CCS_P is strictly more expressive than ABC; explore if CCS_P with $n+1$ priorities is strictly more expressive than CCS_P with n priorities ...

Reading guide In Section 2, some fundamentals are provided, including CLTS, just paths, concurrency closure, etc.

In Section 3, the Component-labelled Transition System (CLTS) semantics for Algebra of Broadcasting Communication (ABC) is proposed, as well as the corresponding definition of well-colored paths. In this section, a detailed proof that CLTS(ABC) respects concurrency closure and that well-coloration is equivalent to justness is shown.

In Section 4, an example is given to argue that the CLTS semantics proposed in Section 3 is not expressive enough to judge a path, which should be considered unjust, to be unjust. Then another version of CLTS semantics for ABC is proposed. More detailed work can be found in [2].

In Section 5, a short proposal of CLTS semantics for CCS_P is shown, with out discussing much details.

In Section 6, the expressiveness of the algebras ABC and CCS with 3 priorities is compared by first showing the semantics of CCS with 3 priorities, then giving an encoding from ABC to CCS_P, then proving that the encoding respects the criterion proposed in [5].

2 Component-labelled Transition Systems

In this section, some basics are presented, in particular (Component-)Labelled Transition System, component, and concurrency. The notations presented here are also used in the following sections.

[1] Johannes.Amanpohjola@data61.csiro.au

Definition 1 (LTS) *A* labelled transition system *(LTS) is a tuple*

$$(S, Tr, source, target, \ell)$$

with S and Tr sets (of states *and* transitions*), source*, *target* : $Tr \to S$ *the source and target state of a transition, and* $\ell : Tr \to \mathscr{L}$ *for some set of transition labels* $\mathscr{L}$*. Sometimes the* labels *are also noted as* actions*.*

Definition 2 (Path) *A* path *in a transition system* $(S, Tr,$ *source*, *target*, $\ell)$ *is an alternating sequence*

$$s_0\, t_1\, s_1\, t_2\, s_2 \cdots$$

of states and transitions, starting with a state and either being infinite or ending with a state, such that $source(t_i) = s_{i-1}$ *and* $target(t_i) = s_i$ *for all relevant i. Sometimes a transition is represented as an arrow, with its label and source, target sets on top of the arrow, like:* $\xrightarrow{l(t)}$*, for a transition t.*

Definition 3 (CLTS) *A* Component-Labelled Transition System *(CLTS) is a tuple* $(TS, \mathscr{C}, npc, afc)$ *with* $TS = (S, Tr, source, target, \ell)$ *an LTS,* $\mathscr{C}$ *a set of* components *and* $npc, afc : Tr \to \mathscr{P}(\mathscr{C})$ *the mappings that associate each* $t \in Tr$ *with sets of* necessary *and* affected *components, and the necessary components being a non-empty set. We represent a transition* $t \in Tr$ *and its source and target by*

$$source(t) \xrightarrow{\ell(t), npc(t), afc(t)} target(t).$$

Definition 4 (Well-colored Path) *Let* $Y \subseteq Act$ *a set of actions (enabled actions). A Y-well-colored path* $\pi = s_1 \xrightarrow{\alpha_1, N_1, A_1} s_2 \xrightarrow{\alpha_2, N_2, A_2} \ldots$ *is a path that has the following property :*

$$\forall k, \forall \alpha \notin Y, \alpha \in Act : (\textit{if } \exists s_k \xrightarrow{\alpha, N, A} t \textit{ then } \exists l \geq k, N \cap A_l \neq \varnothing)$$

with the state s_k *and the transition* $s_l \xrightarrow{\alpha_l, N_l, A_l} s_{l+1}$ *on* π*, the transition* $s_k \xrightarrow{\alpha, N, A} t$ *not on* π*. We note* $\mathscr{WC}(Y)$ *as the set of Y-well-colored paths.*

Informally, a path is well-colored, means if an action is enabled (its execution is possible) by a state on the path, then there must be a transition on the path that affects this action, possibly labelled with this action itself.

Definition 5 (Concurrent) *The transition* $t \in Tr$ *is concurrent with* $u \in Tr$*, notation* $t \smile\!\!\bullet\, u$*, if* $npc(t) \cap afc(u) = \emptyset$*. We note* $t \smile u$ *iff* $t \smile\!\!\bullet\, u$ *and* $u \smile\!\!\bullet\, t$*.*

Informally, $t \smile\!\!\bullet\, u$, means if we are in a state that both t and u are enabled, then the execution of u does not affect the necessary environment that is needed for the execution of t. This leads to the following *concurrency closure* property.

Definition 6 (Concurrency Closure) *If transitions* $t \in Tr$ *such that* $t \not\smile\!\!\bullet\, t$[2] *and* $u \in Tr$ *satisfies* $source(t) = source(u)$ *and* $t \smile\!\!\bullet\, u$*, then there is* $v \in Tr$ *with* $v \not\smile\!\!\bullet\, v$*,* $source(v) = target(u)$*,* $l(v) = l(t)$*, and* $npc(v) = npc(t)$*.*

[2]The condition that a transition is not concurrent with itself is designed to serve CCS with Signals, eliminating the case of signals, and is not very relevant for this report up to now. However, it could be considered to eliminate the influence of broadcasting actions ($\mathscr{B}$! actions) in ABC as well.

3 ABC

3.1 Semantics of ABC

The Algebra of Broadcast Communication (ABC) is an extension of CCS, combined with features of CBS. The following syntax definition is directly taken from chapter 2 of [1].

The Algebra of Broadcast Communication (ABC) is parametrised with sets $\mathscr{A}$ of *agent identifiers*, $\mathscr{B}$ of *broadcast names* and $\mathscr{C}$ of *handshake communication names*; each $A \in \mathscr{A}$ comes with a defining equation $A \stackrel{def}{=} P$ with P being a guarded ABC expression as defined below.

The collections $\mathscr{B}!$ and $\mathscr{B}?$ of *broadcast* and *receive* actions are given by $\mathscr{B}\sharp := \{b\sharp \mid b \in \mathscr{B}\}$ for $\sharp \in \{!,?\}$. The set $\bar{\mathscr{C}}$ of *handshake communication co-names* is $\bar{\mathscr{C}} := \{\bar{c} \mid c \in \mathscr{C}\}$, and the set $\mathscr{H}$ of *handshake actions* is $\mathscr{H} := \mathscr{C} \cup \bar{\mathscr{C}}$, the disjoint union of the names and co-names. The function $\bar{\cdot}$ is extended to $\mathscr{H}$ by declaring $\bar{\bar{c}} = c$.

Finally, $Act := \mathscr{B}! \cup \mathscr{B}? \cup \mathscr{H} \cup \{\tau\}$ is the set of *actions*. Below, B, C range over $\mathscr{A}$, b over $\mathscr{B}$, c over $\mathscr{H}$, η over $\mathscr{H} \cup \{\tau\}$ and α, ℓ over Act. A *relabelling* is a function $f : (\mathscr{B} \to \mathscr{B}) \cup (\mathscr{C} \to \mathscr{C})$. It extends to Act by $f(\bar{c}) = \overline{f(c)}$, $f(b\sharp) = f(b)\sharp$ and $f(\tau) := \tau$. The set Ex_{ABC} of ABC expressions is the smallest set including:

$\mathbf{0}$	*inaction*	$\alpha.\mathrm{P}$	*prefixing*	$\mathrm{P}+Q$	*choice*
$\mathrm{P}\|Q$	*parallel composition*	$\mathrm{P}\backslash c$	*restriction*	$\mathrm{P}[f]$	*relabelling*
B	*agent identifier*	(EXP)			

Losing the *npc*, *afc* sets, the semantics of ABC is then shown in Table 1.

3.2 Semantics of ABC w.r.t. CLTS

3.2.1 Semantics

We extend now the semantics of ABC w.r.t. CLTS in Table 1.

The set of components $\mathscr{C} := \{L, R, \varepsilon\}$, where ε means "everything", and follows the rules: $\forall l \in \mathscr{C}$, $\forall N \subseteq \mathscr{P}(\mathscr{C})$,

$$l.\varepsilon = l \qquad \varepsilon.l = l$$
$$N \cup \{\varepsilon\} = \{\varepsilon\} \quad N \cap \{\varepsilon\} = N$$

Now we proceed to concurrency closure property.

Proposition 1 *The component-labelled operational semantics of ABC respects the concurrency closure property.*

Proof: We will prove by structural induction. Let $\alpha, \beta \in$ Act. Note that for every transition t $\in Tr$, there is $npc(\mathrm{t}) = afc(\mathrm{t})$. Let there be transitions $t_1, t_2, t_3 \in Tr$, $t_1 := P \xrightarrow{\alpha,X,X} P_1$, $t_2 := P \xrightarrow{\beta,Y,Y} P_2$, $t_3 := P_1 \xrightarrow{\beta,Y,Y} P_3$. $X \cap Y = \varnothing$ means $t_2 \smile\!\bullet\, t_1$. The progresses in ABC are constructed recursively from the expressions in (EXP). Let the argument be $\delta(P)$: the process P respects *closure property*. Trivial case: $\delta(\mathbf{0})$ is true.

Suppose $\delta(P)$ is true, we prove by induction that $\delta(S)$ is true, where S is composed from P.

- **Prefixing:** Suppose $S = \alpha.P$, the only possibility is that $X = Y = \{\varepsilon\}$, so we cannot have $X \cap Y = \varnothing$, thus $\delta(S)$ is true.
- **Choice:** Suppose $S = P + Q$, the only possibility is that $X = Y = \{\varepsilon\}$, and $\delta(S)$ is true.
- **Relabelling:** Suppose $S = P[f]$, our two transitions are $t_1 = P[f] \xrightarrow{f(\alpha),X,X} P_1[f]$, $t_2 = P[f] \xrightarrow{f(\ell),Y,Y} P_2[f]$ because $P \xrightarrow{\alpha,X,X} P_1, P \xrightarrow{\ell,Y,Y} P_2$. From $X \cap Y = \varnothing$ we have $P_1 \xrightarrow{\ell,Y,Y} P_3$, because of $\delta(P)$. Then we can compose a transition $P_1[f] \xrightarrow{\ell,Y,Y} P_3[f]$.

Table 1: Structural operational semantics of ABC w.r.t. CLTS

$$\alpha.\mathrm{P} \xrightarrow{\alpha,\{\varepsilon\},\{\varepsilon\}} \mathrm{P} \ (\textsc{Act}) \qquad \frac{\mathrm{P} \xrightarrow{\alpha,N,A} \mathrm{P}'}{\mathrm{P}+Q \xrightarrow{\alpha,\{\varepsilon\},\{\varepsilon\}} \mathrm{P}'} \ (\textsc{Sum-l}) \qquad \frac{Q \xrightarrow{\alpha,N,A} Q'}{\mathrm{P}+Q \xrightarrow{\alpha,\{\varepsilon\},\{\varepsilon\}} Q'} \ (\textsc{Sum-r})$$

$$\frac{\mathrm{P} \xrightarrow{\eta,N,A} \mathrm{P}'}{\mathrm{P}|Q \xrightarrow{\eta,L.N,L.A} \mathrm{P}'|Q} \ (\textsc{Par-l}) \qquad \frac{\mathrm{P} \xrightarrow{c,N,A} \mathrm{P}',\ Q \xrightarrow{\bar{c},Z,W} Q'}{\mathrm{P}|Q \xrightarrow{\tau,L.N\cup R.Z,L.A\cup R.W} \mathrm{P}'|Q'} \ (\textsc{Comm}) \qquad \frac{Q \xrightarrow{\eta,N,A} Q'}{\mathrm{P}|Q \xrightarrow{\eta,R.N,R.A} \mathrm{P}|Q'} \ (\textsc{Par-r})$$

$$\frac{\mathrm{P} \xrightarrow{b\sharp_1,N,A} \mathrm{P}',\ Q \xrightarrow{b?}\!\!\!\!\!\!/}{\mathrm{P}|Q \xrightarrow{b\sharp_1,L.N,L.A} \mathrm{P}'|Q} \ (\textsc{Bro-l}) \qquad \frac{\mathrm{P} \xrightarrow{b\sharp_1,N,A} \mathrm{P}',\ Q \xrightarrow{b\sharp_2,Z,W} Q'}{\mathrm{P}|Q \xrightarrow{b\sharp,L.N\cup R.Z,L.A\cup R.W} \mathrm{P}'|Q'} \ (\textsc{Bro-c})$$

$$\frac{\mathrm{P} \xrightarrow{b?}\!\!\!\!\!\!/,\ Q \xrightarrow{b\sharp_2,N,A} Q'}{\mathrm{P}|Q \xrightarrow{b\sharp_2,R.N,\cup R.A} \mathrm{P}|Q'} \ (\textsc{Bro-r}) \qquad \sharp_1\circ\sharp_2=\sharp\neq- \text{ with } \begin{array}{c|ccc} \circ & ! & ? \\ \hline ! & - & ! \\ ? & ! & ? \end{array}$$

$$\frac{\mathrm{P} \xrightarrow{\ell,N,A} \mathrm{P}'}{\mathrm{P}[f] \xrightarrow{f(\ell),N,A} \mathrm{P}'[f]} \ (\textsc{Rel}) \qquad \frac{\mathrm{P} \xrightarrow{\ell,N,A} \mathrm{P}'}{\mathrm{P}\backslash c \xrightarrow{\ell,N,A} \mathrm{P}'\backslash c} \ (c\neq\ell\neq\bar{c}) \ (\textsc{Res}) \qquad \frac{\mathrm{P} \xrightarrow{\ell,N,A} \mathrm{P}'}{B \xrightarrow{\ell,N,A} \mathrm{P}'} \ (B\overset{def}{=}P) \ (\textsc{Rec})$$

- **Restriction:** Straightforward.
- **Recursion:** Straightforward.
- **Parallel composition:** Suppose $S = P|Q$, our two transitions are $t_1 = P|Q \xrightarrow{\alpha,X,X} P_1|Q_1$, $t_2 = P|Q \xrightarrow{\beta,Y,Y} P_2|Q_2$. The two transitions can take the form of *handshake actions* or *broadcast actions*, put together we have $6*6 = 36$ cases, but the proofs are rather similar. Now we illustrate the idea by proving some of the cases.
  - (Par-l) and (Par-l) Suppose our two transitions are $P|Q \xrightarrow{a,L.X,L.X} P_1|Q$ and $P|Q \xrightarrow{b,L.Y,L.Y} P_2|Q$ $(a,b \in \mathscr{H})$ because $P \xrightarrow{a,X,X} P_1$, $P \xrightarrow{b,Y,Y} P_2$. From $(L.X)\cap(L.Y) = \varnothing$ we can easily derive that $X \cap Y = \varnothing$. Thus by induction we get a transition $P_1 \xrightarrow{b,Y,Y} P_3$, and finally we infer a transition $P_1|Q \xrightarrow{b,L.Y,L.Y} P_3|Q$.
  - (Bro-l) and (Bro-l) Suppose our two transitions are $P|Q \xrightarrow{b\sharp_1,L.X,L.X} P_1|Q$ and $P|Q \xrightarrow{d\sharp_2,L.Y,L.Y} P_2|Q$ $(b\sharp_1, d\sharp_2 \in \mathscr{B}\sharp)$ because $P \xrightarrow{b\sharp_1,X,X} P_1$, $P \xrightarrow{d\sharp_2,Y,Y} P_2$, and $Q \xrightarrow{b?}\!\!\!\!\!\!/$. From $L.X \cap L.Y = \varnothing$ we can easily derive that $X \cap Y = \varnothing$. Thus by induction we get a transition $P_1 \xrightarrow{d\sharp_2,Y,Y} P_3$, and finally we infer a transition $P_1|Q \xrightarrow{d\sharp_2,L.Y,L.Y} P_3|Q$.
  - (Bro-l) and (Bro-r) Suppose our two transitions are $P|Q \xrightarrow{b\sharp_1,L.X,L.X} P_1|Q$ and $P|Q \xrightarrow{d\sharp_2,R.Y,R.Y} P|Q_2$ $(b\sharp_1, d\sharp_2 \in \mathscr{B}\sharp)$ because $P \xrightarrow{b\sharp_1,X,X} P_1$, $Q \xrightarrow{d\sharp_2,Y,Y} Q_2$, which means $Q \xrightarrow{b?}\!\!\!\!\!\!/$, $P \xrightarrow{d?}\!\!\!\!\!\!/$. Then we can easily compose a transition $P_1|Q \xrightarrow{d\sharp_2,R.Y,R.Y} P_3|Q_2$, no matter if P_1 listens to d or not.
  - (Bro-c) and (Bro-c) Suppose our two transitions are $P|Q \xrightarrow{b\sharp',X,X} P_1|Q_1$ and $P|Q \xrightarrow{d\sharp'',Y,Y} P_2|Q_2$ because $P \xrightarrow{b\sharp_1,U_1,U_1} P_1$, $Q \xrightarrow{b\sharp_2,V_1,V_1} Q_1$, $P \xrightarrow{d\sharp_3,U_2,U_2} P_2$, $Q \xrightarrow{d\sharp_4,V_2,V_2} Q_2$, and $\sharp' = \sharp_1\circ\sharp_2 \neq -$, $\sharp'' = \sharp_3\circ\sharp_4 \neq -$. Besides, we have $X = L\cdot U_1 \cup R\cdot V_1$, $Y = L\cdot U_2 \cup R\cdot V_2$. $X \cap Y = \varnothing$ implies that $U_1 \cap U_2 = V_1 \cap V_2 = \varnothing$ therefore by induction we get two transitions $P_1 \xrightarrow{d\sharp_3,U_2,U_2} P_3$ and $Q_1 \xrightarrow{d\sharp_4,V_2,V_2} Q_3$. Then we infer $P_1|Q_1 \xrightarrow{d\sharp'',Y,Y} P_3|Q_3$.

Because the correctness and completeness of structural induction, we have proven that the proposition is true for all processes S in *ABC*, and the CLTS semantics of ABC respects the closure property. □

3.3 Justness

To measure justness, we define *just paths* (by adjusting the classical definition) and *well-colored paths*, and show that these two terms are equivalent. The former gives us a formal description, the latter provides us with sufficient tool to decide conveniently whether a path is just or not.

First we adjust the classical definition of *just paths* to ABC.

Definition 7 (Just paths of ABC w.r.t. CLTS) *For $Y \subseteq \mathscr{H}$[3], the set $\mathscr{J}(Y)$ of Y-just paths is the largest family of predicates such that:*

- *A finite path is Y-just if it ends in a state that admits actions from $Y \cup \mathscr{B}?$ only.*
- *If a path of a process $P|Q$ is Y-just, then it can be decomposed into a Z-just path of P and a Z'-just path of Q such that $Y \supseteq Z \cup Z'$ and $Z \cap \bar{Z}' = \varnothing$.*
- *If a path of a process $P\backslash a$ is Y-just, then it can be decomposed into a $(Y \cup \{a, \bar{a}\})$-just path of P.*
- *If a path of a process $P[f]$ is Y-just, then it can be decomposed into a $f^{-1}(Y)$-just path of P.*
- *If a path of a process P is Y-just and non-empty, then its suffix (the path obtained by removing the very first transition) is also Y-just.*

Consequently, a path is just iff it is $\varnothing$-just.

In the context of ABC, we should note that since Y-just paths are defined over the set Y which is a subset of $\mathscr{H}$, the notation "Z-just path" indicates that Z is also a subset of $\mathscr{H}$.

Also, the definition of well-colored paths does not suffice to characterize just-paths in ABC, thus we need to adjust the definition again. The reasons can be illustrated through the following example:

Example 1 *Consider the process $P := a.A|b?.\mathbf{0}$, $A := a.A$, where $a \in \mathscr{H}$, $b? \in \mathscr{B}?$. The path* [4]

$$P_1 \xrightarrow{a,\{L\}} P_2 \xrightarrow{a,\{L\}} P_3$$

should be characterized as $\{a\}$-just because the end state $P_3 = P$ admits actions from $\{a\} \cup \mathscr{B}?$ only.

However, this path is not $\{a\}$-well-colored according to the definition above. A counterexample can be:

$$\text{for } b? \notin \{a\}, \exists P_1 \xrightarrow{b?,\{R\}} P_2, \text{ but } \nexists l > 1, \text{ such that } N_l \cap \{R\} = \varnothing$$

as every N_l is labeled with (a, $\{L\}$). Thus the well-coloration does not correspond to justness. As a result, we should take the set $\mathscr{B}?$ into consideration when adjusting the definition.

Definition 8 *(Well-colored path w.r.t. ABC) Let $Y \subseteq \mathscr{H}$ a set of actions (enabled actions). A Y-well-colored path $\eta = s_1 \xrightarrow{\alpha_1,N_1,A_1} s_2 \xrightarrow{\alpha_2,N_2,A_2} \ldots$ is a path that has the following property :*

$$\forall k, \forall \alpha \notin (Y \cup \mathscr{B}?), \alpha \in Act : (if \exists s_k \xrightarrow{\alpha,N,A} t \Longrightarrow \exists l \geq k, N \cap A_l \neq \varnothing)$$

for s_k, $s_l \xrightarrow{\alpha_l,N_l,A_l}$ on π, $s_k \xrightarrow{\alpha,N,A} t$ not on π. We note $\mathscr{WC}(Y)$ as the set of Y-well-colored paths.

Now we proceed to prove that these two definitions are equivalent. [5]

[3] It can be considered whether to restrict Y as a subset of $\mathscr{H}$ or define Y as a subset of Act, i.e. not ruling the set $\mathscr{B}!$ out.

[4] Because the npc and afc sets are always the same in this version of CLTS semantics for ABC, we sometimes leave out the second set in the notation.

[5] The proof is almost identical with the proof in Victor(victor.dyseryn-fostier@polytechnique.edu)'s report, only adding some details.

Proposition 2 *A path in ABC(CLTS) is Y-just iff it is Y-well-colored, which means* $\mathscr{WC}(Y) = \mathscr{J}(Y)$.

Proof: *Part 1: "*$\mathscr{WC}(Y) \subseteq \mathscr{J}(Y)$*"*

- All finite Y-well-colored path are also Y-just. Let s_k be the end state, then we have the proof.
- If a path of a process P|Q is Y-well-colored, then (I)it can be decomposed into a Z-just path of P and a Z'-just path of Q such that (II)$Y \supseteq Z \cup Z'$ and (III)$Z \cap \bar{Z}' = \varnothing$.

  Let the path be π, and its decomposition are π_1 and π_2, and

  $$\pi_1 = s_1 \xrightarrow{\alpha_1, N_1} s_2 \xrightarrow{\alpha_2, N_2} \ldots$$

  $$\pi_2 = s'_1 \xrightarrow{\alpha'_1, N'_1} s'_2 \xrightarrow{\alpha'_2, N'_2} \ldots$$

  [6] with $\alpha_i, \alpha'_i \in Act$, $s_i|s'_i$ the states on π, $\forall i$. Note that, a transition on π can take the form of interleaving, for example, in the case of left interleaving, $s_i|s'_i \xrightarrow{\alpha_i, X_i} s_{i+1}|s'_{i+1}$ is decomposed as $s_i \xrightarrow{\alpha_i, N_i} s_{i+1}$ on π_1, with $X_i = L.N_i$, and nothing on π_2. For the reason of symmetry, we note a transition $s'_i \xrightarrow{\lambda, N'_i} s'_{i+1}$ on π_2, where $N'_i = \varnothing$, $s'_i = s'_{i+1}$, and $\lambda \notin Act$ a notation emphasizing that the transition does not really exist. Then we always have $X_i = L.N_i \cup R.N'_i$, because $R.\varnothing = \varnothing$. In case of right interleaving, our notation changes symmetrically.

  Now we define two sets Z and Z', where $\mathscr{H} \supseteq Z, Z'$,and show that π_1 is Z-well-colored and π_2 is Z'-well-colored. Let

  $$Z := \{a \mid \exists k,\ s_k \xrightarrow{a, N} t \text{ and } \forall l \geq k, N_l \cap N = \varnothing\}$$

  $$Z' := \{a \mid \exists k,\ s'_k \xrightarrow{a, N'} {}' t' \text{ and } \forall l \geq k, N'_l \cap N' = \varnothing\}$$

  We can see easily from the definition that the path π_1 is Z-well-colored and π_2 is Z'-well-colored, as described in (I).

  Now we prove that $Y \supseteq Z$ and $Y \supseteq Z'$.

  - Assume $\exists a \in \mathscr{H}$ such that $a \in Z \backslash Y$. Then we have a transition in π starting from a state s_k on π, composed from the corresponding transitions starting with states in π_1 and π_2:

    $$s_k|s'_k \xrightarrow{a, L.N} t|s'_k$$

    where $\forall l > k, N_l \cap N = \varnothing$ because $a \in Z$, and $\exists l > k, X_l \cap L.N \neq \varnothing$, $X_l = L.N_l \cup R.N'_l$, because $a \notin Y$. Then we have $(L.N_l \cup R.N'_l) \cap L.N \neq \varnothing$, thus $(L.N_l \cap L.N) \cup (R.N'_l \cap L.N) \neq \varnothing$, and $(L.N_l \cap L.N) \neq \varnothing$, then $(N_l \cap N) \neq \varnothing$, which leads to contradiction. Thus $\nexists a \in Z \backslash Y$, which means $Y \supseteq Z$.
  - Analogically we can prove that $Y \supseteq Z'$.

  Then we can have $Y \supseteq Z \cup Z'$, and (II) is proven.

  Now we proceed to prove $Z \cap \bar{Z}' = \varnothing$.

  Assume we have a $c \in Z \cap \bar{Z}'$, which means $\exists s_m \xrightarrow{c, N} t$, $\forall l \geq m, N_l \cap N = \varnothing$, and $\exists s'_n \xrightarrow{\bar{c}, N'} t'$, $\forall l \geq n$, $N'_l \cap N' = \varnothing$. Without losing generality, we can assume that $m \leq n$. Because of concurrency closure, we have a transition $s_n \xrightarrow{c, N} r$, $\forall l \geq n$, $N_l \cap N = \varnothing$. Then we can compose a transition $s_n|s'_n \xrightarrow{\tau, L.N \cup R.N'} r|t$, $\forall l \geq n$, $N_l \cap N = \varnothing$ and $N'_l \cap N' = \varnothing$ with $s_n|s'_n$ on π.

  Because $\tau \notin Y$, $\exists h \geq n$, $X_h \cap (L.N \cup R.N') \neq \varnothing$, $X_h = L.N_h \cup R.N'_h$. Therefore we have $(L.N_h \cap L.N) \cup (L.N_h \cap R.N') \cup (R.N'_h \cap L.N) \cup (R.N'_h \cap R.N') \neq \varnothing$, which again leads to contradiction. Thus $\nexists c$, and $Z \cap \bar{Z}' = \varnothing$, (III) is proven.

[6] Here we leave out the second sets again, for they are identical with the first sets.

- If a path of a process $P\backslash a$ is Y-well-colored, then it can be decomposed into a $(Y \cup \{a, \bar{a}\})$-well-colored path of P. A decomposition of a transition $P\backslash a \xrightarrow{b,N} P'\backslash a$ is the transition $P \xrightarrow{b,N} P'$ where $b \neq a \neq \bar{a}$. Let the path be π, and its decomposition be π'. We note the paths as:

$$\pi := s_1\backslash a \xrightarrow{b_1,N_1} s_2\backslash a \xrightarrow{b_2,N_2} s_3\backslash a...$$

$$\pi' := s_1 \xrightarrow{b_1,N_1} s_2 \xrightarrow{b_2,N_2} s_3...$$

  $\forall c \notin Y \cup \{a, \bar{a}\}$, such that $\exists s_k \xrightarrow{c,N} t$, because $c \neq a \neq \bar{a}$, there exists a transition $s_k\backslash a \xrightarrow{c,N} t$. And because $c \notin Y$, there exists a transition $s_l\backslash a \xrightarrow{b_l,N_l} s_{l+1}\backslash a, l \geq k$, such that $N_l \cap N \neq \varnothing$, because π is Y-well-colored. Then we can decompose the transition into $s_l \xrightarrow{b_l,N_l} s_{l+1}$, $l \geq k$, $N_l \cap N \neq \varnothing$. Then we have proven that π' is $Y \cup \{a, \bar{a}\}$-well-colored.

- If a path of a process $P[f]$ is Y-well-colored, then it can be decomposed into a $f^{-1}(Y)$-well-colored path of P. Let the paths be:

$$\pi := s_1 \xrightarrow{\alpha_1,N_1} s_2 \xrightarrow{\alpha_2,N_2} s_3...$$

$$\pi' := s_1^{-1}[f] \xrightarrow{f^{-1}(\alpha_1),N_1} s_2^{-1}[f] \xrightarrow{f^{-1}(\alpha_2),N_2} s_3^{-1}[f]...$$

  Because of the semantics of relabelling, we can transform the conditions

$$\forall c \notin Y, \text{ such that } \exists s_k \xrightarrow{c,N} t, \text{ then } \exists s_l \xrightarrow{\alpha_l,N_l} s_{l+1}, l \geq k$$

  into

$$\forall c \notin Y^{-1}[f], \text{ such that } \exists s_k^{-1}[f] \xrightarrow{c,N} t, \text{ then } \exists s_l^{-1}[f] \xrightarrow{f^{-1}(\alpha_l),N_l} s_{l+1}^{-1}[f], l \geq k$$

  Thus we have that π' is $f^{-1}(Y)$-well-colored.

- If a path of a process P is Y-well-colored and non-empty, then its suffix (the path obtained by removing the very first transition) is also Y-well-colored. The definition gives immediately that if a path is Y-well-colored and non-empty, then its suffix is Y-well-colored.

Part 2: "$\mathscr{J}(Y) \subseteq \mathscr{WC}(Y)$" (By induction.)

We write that a transition is of depth M when it applies M-times the rules in Table 1. For example, $a.\mathbf{0} \xrightarrow{a,\{\varepsilon\}} \mathbf{0}$ has depth 0, $(a.P|b.Q)|c.R \xrightarrow{a,\{LL\}} (P|b.Q)|c.R$ has depth 2, because:

$$\dfrac{\dfrac{a.P \xrightarrow{a,\{\varepsilon\}} P}{a.P|b.Q \xrightarrow{a,\{L\}} P|b.Q}}{(a.P|b.Q)|c.R \xrightarrow{a,\{LL\}} (P|b.Q)|c.R}$$

Let property $¶(M)$ ($M \in \mathbb{N}$) be: $\forall Y \in \mathscr{H}$, for all Y-just paths $\pi = s_1 \xrightarrow{\alpha_1,N_1} s_2 \xrightarrow{\alpha_2,N_2} ...$, $\forall k$, $\forall \alpha \notin (Y \cup \mathscr{B}?)$, if $\exists s_k \xrightarrow{\alpha,N} t$ of depth $\leq M$, then $\exists l \geq k$ such that $N_l \cap N \neq \varnothing$.

Note that s_k cannot be the end state. Assume s_k is the end state, then we cannot have such transition $s_k \xrightarrow{\alpha,N} t$ because π is Y-just and $\alpha \notin Y$.

- $¶(0)$ is true. If $\exists s_k \xrightarrow{\alpha,N} t$ of depth 0, then it is only possible that s_k is in the form $\alpha.P$, where P is a process. Then s_k is the only possible process outgoing from s_k, and therefore $s_k \xrightarrow{\alpha,N} t$ must be on π. And it contradicts with the condition that $s_k \xrightarrow{\alpha,N} t$ not on π.

- If $¶(M)$ is true then $¶(M+1)$ is true. We now focus on the property $¶(M+1)$ and analyse the cases of different derivations from Table 1.

  - (SUM-L), (SUM-R) $s_k = P_1 + P_2$, then we have $N = \{\varepsilon\}$ and $N \cap N_l = N_l \neq \varnothing$, because in ABC, the *npc*, *afc* sets are always non-empty.

- (PAR-L) Suppose $s_k = P|P'$, $t = R|P'$ and $N = L.U$ because $P \xrightarrow{a,U} R$. We decompose the path starting from s_k: $s_k = P|P' \xrightarrow{a_k,N_k} s_{k+1}$..., which is Y-just, into two paths: $\eta = P \xrightarrow{b_k,U_k} ...$ and $\eta' = P' \xrightarrow{b'_k,U'_k} ...$ which are respectively Z and Z'-just such that $Z \cup Z' \subseteq Y$ and $Z \cap \bar{Z}' = \varnothing$. η is a Z-just path, $a \notin Z$ because $a \notin Y$ and $Z \subseteq Y$, and the transition $P \xrightarrow{a,U} R$ is of depth M so by $¶(M)$ there exists $l \geq k$ such that $U_l \cap U \neq \varnothing$. As a result $L.U_l \cap L.U \neq \varnothing$ and since $L.U_l \subseteq N_l$ and $N = L.U$, $N_l \cap N \neq \varnothing$.
- (PAR-R), (COMM), (BRO-L), (BRO-R), (BRO-C) Similar to (PAR-L).
- (REC) Suppose $s_k = A := P$ and the transition $s_k \xrightarrow{\alpha,N} t$ has depth $m+1 \leq M+1$, then the transition $P \xrightarrow{\alpha,N} t$ has depth $m \leq M$. Replacing s_k with P we acquire a path π', which is also Y-just. Because $¶(M)$ is true, we have an $l \geq k$ on π' such that $N_l \cap N \neq \varnothing$. It is obvious that $s_l \xrightarrow{\alpha_l,N_l} s_{l+1}$ is also on π, thus $¶(M+1)$ is also true.
- (RES) Similar to (REC).
- (REL) Suppose $s_k = P[f]$ for a relabelling function f and a process P. The transition $P[f] \xrightarrow{f(\alpha),N} t[f]$ has depth $m+1 \leq M+1$, then the transition $P \xrightarrow{\alpha,N} t$ has depth $m \leq M$. We decompose again the path starting from s_k, and like before, we can find a transition that is not concurrent to $s_k \xrightarrow{\alpha,N} t$, thus the corresponding transition is not concurrent to $P[f] \xrightarrow{f(\alpha),N} t[f]$.

Because of the correctness and completeness of induction, the property $¶(M)$ is true for all $M \in \mathbb{N}$, and we have $\mathscr{J}(Y) = \mathscr{WC}(Y)$.

□

4 Another proposal: CLTS for ABC

In this section, we first explain a counterexample that shows the inadequacy with respect to justness of the CLTS proposed above, then propose another version of CLTS for ABC.

Example 2 *Consider the process $P := b!.\mathbf{0}|b?.\mathbf{0} + X$ where $X := \tau.X$. Then we can compose a path according to Section 3:*

$$P \xrightarrow{\tau,\{R\},\{R\}} P \xrightarrow{\tau,\{R\},\{R\}} P \xrightarrow{\tau,\{R\},\{R\}} P \xrightarrow{\tau,\{R\},\{R\}} \dots$$

where P continues doing τ-loop and returning to the initial state.

This path would be considered just because each of the transitions effect the components that is necessary for the execution of $b!$, which is $\{L,R\}$. However, this path should not be considered as just, because the parallel component $b!.\mathbf{0}$ is enabled at each state, and yet never executed, which contradicts to the fact that a $\mathscr{B}!$ action should be non-blocking. Also, it should be considered just according to Appendix C in [1].

For the purpose of solving this problem, we propose another approach of defining CLTS semantics for ABC. For more please refer to [2].

5 CCS with priority

CCS with priority(CCS_P) is an extension of CCS, assigning each action (label in LTS) with a certain priority. CCS_P can be useful, for example, when building systems containing components that has interrupting power, for example the occurrence of an error.

Table 2: Structural operational semantics of CCS_P

$$\alpha.\mathrm{P} \xrightarrow{\alpha} \mathrm{P} \text{ (ACT)}$$

$$\frac{\mathrm{P} \xrightarrow{\underline{a}} \mathrm{P}'}{\mathrm{P}+Q \xrightarrow{\underline{a}} \mathrm{P}'} \text{ (SUM-LH)} \qquad \frac{Q \xrightarrow{\underline{a}} Q'}{\mathrm{P}+Q \xrightarrow{\underline{a}} Q'} \text{ (SUM-RH)}$$

$$\frac{\mathrm{P} \xrightarrow{a} \mathrm{P}',\ Q \in S_p}{\mathrm{P}+Q \xrightarrow{a} \mathrm{P}'} \text{ (SUM-LL)} \qquad \frac{Q \xrightarrow{a} Q',\ \mathrm{P} \in S_p}{\mathrm{P}+Q \xrightarrow{a} Q'} \text{ (SUM-RL)}$$

$$\frac{\mathrm{P} \xrightarrow{\underline{a}} \mathrm{P}'}{\mathrm{P}|Q \xrightarrow{\underline{a}} \mathrm{P}'|Q} \text{ (PAR-LH)} \qquad \frac{Q \xrightarrow{\underline{a}} Q'}{\mathrm{P}|Q \xrightarrow{\underline{a}} \mathrm{P}|Q'} \text{ (PAR-RH)}$$

$$\frac{\mathrm{P} \xrightarrow{a} \mathrm{P}',\ Q \in S_p}{\mathrm{P}|Q \xrightarrow{a} \mathrm{P}'|Q} \text{ (PAR-LL)} \qquad \frac{Q \xrightarrow{a} Q',\ \mathrm{P} \in S_p}{\mathrm{P}|Q \xrightarrow{a} \mathrm{P}|Q'} \text{ (PAR-RL)}$$

$$\frac{\mathrm{P} \xrightarrow{\underline{c}} \mathrm{P}',\ Q \xrightarrow{\bar{\underline{c}}} Q'}{\mathrm{P}|Q \xrightarrow{\underline{\tau}} \mathrm{P}'|Q'} \text{ (COMM-H)} \qquad \frac{\mathrm{P} \xrightarrow{c} \mathrm{P}',\ Q \xrightarrow{\bar{c}} Q',\ \mathrm{P}|Q \in S_p}{\mathrm{P}|Q \xrightarrow{\tau} \mathrm{P}'|Q'} \text{ (COMM-L)}$$

$$\frac{\mathrm{P} \xrightarrow{\ell} \mathrm{P}'}{\mathrm{P}[f] \xrightarrow{f(\ell)} \mathrm{P}'[f]} \text{ (REL)} \qquad \frac{\mathrm{P} \xrightarrow{\ell} \mathrm{P}'}{\mathrm{P}\backslash c \xrightarrow{\ell} \mathrm{P}'\backslash c}\ (c \neq \ell \neq \bar{c}) \text{ (RES)} \qquad \frac{\mathrm{P} \xrightarrow{\ell} \mathrm{P}'}{B \xrightarrow{\ell} \mathrm{P}'}\ (B \overset{def}{=} P) \text{ (REC)}$$

We start with CCS with two priorities. It is argued in [3] that the two priorities can be easily extended to discrete priorities. The basic idea is: actions with high priority cannot be preempted by other actions, while actions with low priority can be preempted by silent(internal) action with high priority. As usual, we define the following notations:

CCS_P is parametrised with sets $\mathscr{A}$ of *agent identifiers*, $\mathscr{C}_l$ of *communication names with low priority*, $\mathscr{C}_h$ of *communication names with high priority*; each $A \in \mathscr{A}$ comes with a defining equation $A \overset{def}{=} P$ with P being a guarded CCS_P expression as defined below.

The set $\mathscr{H}_l$ of *handshake actions with low priority* is $\mathscr{H}_l := \mathscr{C}_l \cup \bar{\mathscr{C}}_l$, $\mathscr{H}_h$ of *handshake actions with high priority* is $\mathscr{H}_h := \mathscr{C}_h \cup \bar{\mathscr{C}}_h$, the disjoint union of the names and co-names. The function $\bar{\cdot}$ is extended to $\mathscr{H}_l$ and $\mathscr{H}_h$ by declaring $\bar{\bar{c}} = c$ for c$\in \mathscr{C}_l$ or c$\in \mathscr{C}_h$.

Finally, $Act := \mathscr{H}_l \cup \mathscr{H}_h \cup \{\tau\} \cup \{\underline{\tau}\}$ is the set of *actions* where $\underline{\tau}$ is the silent action with high priority. Below, B, C range over $\mathscr{A}$, a over $\mathscr{H}_l \cup \{\tau\}$, c over $\mathscr{H}_l$, $\underline{a}$ over $\mathscr{H}_h \cup \{\underline{\tau}\}$, $\underline{c}$ over $\mathscr{H}_h$ and α, ℓ over Act. A *relabelling* is a function $f : (\mathscr{C}_l \to \mathscr{C}_l) \cup (\mathscr{C}_h \to \mathscr{C}_h)$. It extends to Act by $f(\bar{c}) = \overline{f(c)}$, and $f(\tau) := \tau$, $f(\underline{\tau}) := \underline{\tau}$. The set $S_{\mathrm{CCS_P}}$ (*State*) of CCS_P expressions is the smallest set including:

$\mathbf{0}$	*inaction*	$\alpha.\mathrm{P}$	*prefixing*	$\mathrm{P}+Q$	*choice*
$\mathrm{P}\|Q$	*parallel composition*	$\mathrm{P}\backslash c$	*restriction*	$\mathrm{P}[f]$	*relabelling*
A	*agent identifier*				

We note $S_p \subseteq S_{\mathrm{ABC}}$ as the set of *patient processes* (*states*). A process is *patient* if it does not do *internal action with high priority*. Formally, $P \in S_p$ iff

$$\forall \alpha \text{ such that } \exists P \xrightarrow{\alpha} P',\ \{\alpha\} \cap \{\underline{\tau}\} = \varnothing$$

The semantics of CCS_P is defined as in Table 2. It is equivalent as in [3], but defined in one stage instead of two stages, and negative premises are avoided.

Adding *component* to the semantics, we have Table 3.

Table 3: Structural operational semantics of CCS_P w.r.t. CLTS

$$\alpha.\mathrm{P} \xrightarrow{\alpha,\{\varepsilon\},\{\varepsilon\}} \mathrm{P} \text{ (ACT)}$$

$$\frac{\mathrm{P} \xrightarrow{a,N,A} \mathrm{P}'}{\mathrm{P}+Q \xrightarrow{a,\{\varepsilon\},\{\varepsilon\}} \mathrm{P}'} \text{ (SUM-LH)} \qquad \frac{Q \xrightarrow{a,N,A} Q'}{\mathrm{P}+Q \xrightarrow{a,\{\varepsilon\},\{\varepsilon\}} Q'} \text{ (SUM-RH)}$$

$$\frac{\mathrm{P} \xrightarrow{a,N,A} \mathrm{P}',\ Q \in S_p}{\mathrm{P}+Q \xrightarrow{a,\{\varepsilon\},\{\varepsilon\}} \mathrm{P}'} \text{ (SUM-LL)} \qquad \frac{Q \xrightarrow{a,N,A} Q',\ \mathrm{P} \in S_p}{\mathrm{P}+Q \xrightarrow{a,\{\varepsilon\},\{\varepsilon\}} Q'} \text{ (SUM-RL)}$$

$$\frac{\mathrm{P} \xrightarrow{a,N,A} \mathrm{P}'}{\mathrm{P}|Q \xrightarrow{a,L.N,L.A} \mathrm{P}'|Q} \text{ (PAR-LH)} \qquad \frac{Q \xrightarrow{a,N,A} Q'}{\mathrm{P}|Q \xrightarrow{a,R.N,R.A} \mathrm{P}|Q'} \text{ (PAR-RH)}$$

$$\frac{\mathrm{P} \xrightarrow{a,N,A} \mathrm{P}',\ Q \in S_p}{\mathrm{P}|Q \xrightarrow{a,L.N,L.A} \mathrm{P}'|Q} \text{ (PAR-LL)} \qquad \frac{Q \xrightarrow{a,N,A} Q',\ \mathrm{P} \in S_p}{\mathrm{P}|Q \xrightarrow{a,R.N,R.A} \mathrm{P}|Q'} \text{ (PAR-RL)}$$

$$\frac{\mathrm{P} \xrightarrow{c,N,A} \mathrm{P}',\ Q \xrightarrow{\bar{c},Z,W} Q'}{\mathrm{P}|Q \xrightarrow{\tau,L.N\cup R.Z,L.A\cup R.W} \mathrm{P}'|Q'} \text{ (COMM-H)} \qquad \frac{\mathrm{P} \xrightarrow{c,N,A} \mathrm{P}',\ Q \xrightarrow{\bar{c},Z,W} Q',\ \mathrm{P}|Q \in S_p}{\mathrm{P}|Q \xrightarrow{\tau,L.N\cup R.Z,L.A\cup R.W} \mathrm{P}'|Q'} \text{ (COMM-L)}$$

$$\frac{\mathrm{P} \xrightarrow{\ell,N,A} \mathrm{P}'}{\mathrm{P}[f] \xrightarrow{f(\ell),N,A} \mathrm{P}'[f]} \text{ (REL)} \qquad \frac{\mathrm{P} \xrightarrow{\ell,N,A} \mathrm{P}'}{\mathrm{P}\backslash c \xrightarrow{\ell,N,A} \mathrm{P}'\backslash c}\ (c\neq\ell\neq\bar{c}) \text{ (RES)} \qquad \frac{\mathrm{P} \xrightarrow{\ell,N,A} \mathrm{P}'}{B \xrightarrow{\ell,N,A} \mathrm{P}'}\ (B \stackrel{def}{=} P) \text{ (REC)}$$

Proposition 3 *The component-labelled operational semantics of CCS_P respects the concurrency closure property.*

Proof: Similar like in Section 3. □

Definition 9 (Just paths of CCS_P w.r.t. CLTS) *For $Y \subseteq (\mathscr{H}_l \cup \mathscr{H}_h)$, the set $\mathscr{J}(Y)$ of Y-just paths is the largest family of predicates such that:*

- *A finite path is Y-just if it ends in a state that admits actions from Y only.*
- *If a path of a process $P|Q$ is Y-just, then it can be decomposed into a Z-just path of P and a Z'-just path of Q such that $Y \supseteq Z \cup Z'$ and $Z \cap \bar{Z}' = \varnothing$.*
- *If a path of a process $P\backslash a$ is Y-just, then it can be decomposed into a $(Y \cup \{a, \bar{a}\})$-just path of P.*
- *If a path of a process $P[f]$ is Y-just, then it can be decomposed into a $f^{-1}(Y)$-just path of P.*
- *If a path of a process P is Y-just and non-empty, then its suffix (the path obtained by removing the very first transition) is also Y-just.*

Consequently, a path is just iff it is $\varnothing$-just.

Definition 10 *(Well-colored path w.r.t. CCS_P) Let $Y \subseteq (\mathscr{H}_l \cup \mathscr{H}_h)$ a set of actions (enabled actions). A Y-well-colored path $\eta = s_1 \xrightarrow{\alpha_1,N_1,A_1} s_2 \xrightarrow{\alpha_2,N_2,A_2} \ldots$ is a path that has the following property :*

$$\forall k, \forall \alpha \notin Y, \alpha \in Act : (if \exists s_k \xrightarrow{\alpha,N,A} t \Longrightarrow \exists l \geq k, N \cap A_l \neq \varnothing)$$

for $s_k, s_l \xrightarrow{\alpha_l,N_l,A_l}$ on π, $s_k \xrightarrow{\alpha,N,A} t$ not on π. We note $\mathscr{WC}(Y)$ as the set of Y-well-colored paths.

Proposition 4 *A path in ABC(CLTS) is Y-just iff it is Y-well-colored, which means $\mathscr{WC}(Y) = \mathscr{J}(Y)$.*

Proof: Similar like in Section 3. □

6 Expressiveness of ABC and CCS with 3 Priorities

In this section, we compare the relative expressiveness of the process algebras ABC and CCS with 3 Priorities, using criteria proposed in [5]. We start with comparing ABC and CCS with 3 priorities, finding an encoding from ABC to CCS with 3 Priorities that respects the encodability criteria, then prove by separation results that there is no suitable encoding from CCS with 3 Priorities to ABC.

We expand now CCS with 3 Priorities introduced in Section 5 with another priority, namely *medium*. Now the operational semantics change as follows: informally, a process can execute an action with *low* priority, when the process does no silent action with *medium* or *high* priority; a process can execute an action with *medium* priority, when the process does no silent action with *high* priority; actions with *high* priority will not be preempted by any *silent actions*. The *Structural Operation Semantics* of this process algebra can be easily acquired by extending Table 3.

Let Act_{ABC} be the set of labels of ABC, Act_{CCS_P} the set of labels in CCS with 3 Priorities, $\mathscr{C}_m$ the set of *communication names* with *medium* priority, $\mathscr{H}_m := \mathscr{C}_m \cup \bar{\mathscr{C}}_m$, $Act := \mathscr{H}_l \cup \mathscr{H}_m \cup \mathscr{H}_h \cup \{\tau_l\} \cup \{\tau_m\} \cup \{\tau_h\}$. τ_l, τ_m, τ_h are the silent actions with *low*, *medium*, and *high* priority, respectfully.

Below, a_l over $\mathscr{H}_l \cup \{\tau_l\}$, c_l over $\mathscr{H}_l$, a_m over $\mathscr{H}_m \cup \{\tau_m\}$, c_m over $\mathscr{H}_m$, a_h over $\mathscr{H}_h \cup \{\underline{\tau_h}\}$, c_h over $\mathscr{H}_h$ and α, ℓ over Act, $\tau \in \{\tau_l, \tau_m, \tau_h\}$. The other notations are the same as in Section 5.

6.1 Encode ABC to CCS$_P$

The following encoding is proposed by Johannes Åman Pohjola[7].

The injective function i: $Act_{ABC} \to A_{CCS_P}$ is a mapping of actions in ABC to names in CCS_P, trivially,

$$i(\mathbf{0}) = \mathbf{0},\ i(\bar{\eta}) = \overline{i(\eta)} (\eta \in \mathscr{H}),\ i(\tau) = \tau$$

and priorities are assigned for each $i(\cdot)$ by suffixes.

$[\![\cdot]\!]$: $S_{ABC} \to S_{CCS_P}$ is the encoding from ABC to CCS_P, $P, Q \in S_{ABC}$. The encoding is as follows:
$[\![b!.P]\!] := \tau_l.(\overline{i(b?)}_h.X + \tau_m.[\![P]\!])$ where $X := i(b?)_h.X + \tau_m.[\![P]\!]$;
$[\![b?.P]\!] := i(b?)_h.\tau_m.[\![P]\!]$;
$[\![\eta.P]\!] := i(\eta)_l.[\![P]\!]$;
$[\![P|Q]\!] := [\![P]\!]|[\![Q]\!]$;
$[\![P+Q]\!] := [\![P]\!] + [\![Q]\!]$;
[8]

6.2 Adequacy of the Encoding

Now we prove that the encoding respects the following criteria, we only give the definitions of the criteria here, for more details, please refer to [5].

Let L_1, L_2 be calculi, $S, S_1, \ldots, S_k$ be subterms, N the set of *names*, F_N be *free names* of the subterms, σ a subsititution of names for names, $\varphi_{[\![]\!]}$ a renaming policy, $\asymp$ a behavioral equivalence, $\Mapsto$ the reflexive and transitive closure of $\xrightarrow{\alpha}$ where $\alpha = \tau$ if the transition is in CCS$_P$, or α is a silent or broadcast action if the transition is an ABC transition. The suffixes $_1$ and $_2$ assigns each relation to ABC or CCS$_P$.

[7] Johannes.Amanpohjola@data61.csiro.au

[8] Is it necessary to encode *relabelling*, *restriction*, *recursion*?

6.2.1 Compositionality

Property 1 (Compositionality) *A translation $[\![\cdot]\!] : L_1 \to L_2$ is compositional if, for every k-ary operator op of L_1 and for every subset of names N, there exists a k-ary context $C^N_{op}(\_1;\ldots;\_k)$ such that, for all $S_1,\ldots,S_k$ with $F_N(S_1,\ldots,S_k) = N$, it holds that $[\![op(S_1,\ldots,S_k)]\!] = C^N_{op}([\![S_1]\!];\ldots;[\![S_k]\!])$.*

Proof: The compositionality guarantees that the translation of a compound term must be defined in terms of the translation of the subterms, in which way our encoding is defined. □

6.2.2 Name invariance

Property 2 (Name invariance) *A translation $[\![\cdot]\!] : L_1 \to L_2$ is name invariant if, for every S and σ, it holds that*

$$[\![S\sigma]\!] \begin{cases} = [\![S]\!]\sigma' & \text{if } \sigma \text{ is injective} \\ \asymp_2 [\![S]\!]\sigma' & \text{otherwise} \end{cases}$$

Where σ' is such that $\varphi_{[\![]\!]}(\sigma(a)) = \sigma'(\varphi_{[\![a]\!]})$ for every $a \in N$.

Proof: Naturally, because the encoding does no name variations. □

6.2.3 Operational correspondence

Property 3 (Operational correspondence) *A translation $[\![\cdot]\!] : L_1 \to L_2$ is operational correspondent if it is*

- (COMPLETE)*: for all $S \Longmapsto_1 S'$, it holds that $[\![S]\!] \Longmapsto_2 \asymp_2 [\![S']\!]$;*
- (SOUND)*: for all $[\![S]\!] \Longmapsto_2 T$, there exists an S' such that $S \Longmapsto_1 S'$ and $T \Longmapsto_2 \asymp_2 [\![S']\!]$.*

Proposition 5 *The encoding from ABC to CCS_P respects operational correspondence property up to* strong bisimulation.

Definition 11 (Strong bisimulation) *We note $s_1 \in LTS_1$, $s_2 \in LTS_2$, $\alpha \in Act$.*

A relation $\mathscr{B} \subseteq (LTS_1 \times LTS_2 \cup LTS_2 \times LTS_1)$ is a strong bisimulation *if:*

i) For all $s_1 \mathscr{B} s_2$, for all α such that $s_1 \xrightarrow{\alpha} s_1'$, there exists $s_2 \xrightarrow{\alpha} s_2'$ such that $s_1 \mathscr{B} s_2$.

ii) $\mathscr{B}$ is symmetric.

Strong bisimulation is congruent to all the operators in ABC and CCS_P. We note $s_1 \asymp s_2$ if there exists a strong bisimulation $\mathscr{B}$ such that $s_1 \mathscr{B} s_2$

Obviously, the encoding respects the (COMPLETE) property if $S = S'$, because the relation $\Longmapsto$ is reflexive. Then we only need to discuss the case where $S \neq S'$. In this subsection, we always denote $\to_1$ as transitions that are labelled with *silent* or *broadcast* actions, and $\to_2$ as labelled with *silent* actions.

To make the proof more clear, we need to first build some lemmas and prove them to be true.

Lemma 1 *For all $S \to_1 S'$, it holds that $[\![S]\!] \Longmapsto_2 \asymp_2 [\![S']\!]$.*

Its proof comes later. This lemma helps us simplify the cases how S' is reached from S, reducing to only one transition.

Lemma 2 *The encoding from ABC to CCS_P respects* (COMPLETE) *property up to* strong bisimulation *if Lemma 1 is proven.*

Proof: Suppose Lemma 1 is true, $S \Longmapsto_1 S'$ through the path

$$S \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_n} S'$$

where s_i are all ABC processes, and α_i are all *silent* or *broadcast* actions, for relevant $i \in \mathbb{N}$. Because of Lemma 1 we have $[\![S]\!] \Longmapsto_2 t_1$, where t_1 is strongly bisimilar to $[\![s_1]\!]$. Because $s_1 \xrightarrow{\alpha_2} s_2$, we have $[\![s_1]\!] \Longmapsto_2 t_2$, where t_2 is strongly bisimilar to $[\![s_2]\!]$. Because $t_1 \asymp_2 [\![s_1]\!]$, we have $t_1 \Longmapsto_2 t_2$ due to bisimulation. Like this, we can construct a path

$$[\![S]\!] \Longmapsto_2 t_1 \Longmapsto_2 t_2 \Longmapsto_2 \cdots \Longmapsto_2 T$$

where $t_i \asymp_2 [\![s_i]\!]$ for all relevant i, and $T \asymp_2 [\![S']\!]$. Thus we have $[\![S]\!] \Longmapsto_2 \asymp_2 [\![S']\!]$, and the *completeness* criterion is held. □

To prove Lemma 1, we need other lemmas:

Lemma 3 *If P is an ABC process, then $[\![P]\!]$ has no preempting power.*

Proof: From the encoding we can see: $[\![P]\!]$ is always guarded with either τ with *low* priority, or an action that is not silent at all. Since only τ with *medium* or *high* priority has the preempting power, we can see that $[\![P]\!]$ has no preempting power. □

Lemma 4 *If P is an ABC process, for all $P \xrightarrow{c} P'$ where $c \in \mathscr{H}$, it holds that $[\![P]\!] \xrightarrow{i(c)_l} [\![P']\!]$.*

Proof: We prove by induction.
Let the assumption be: $\delta(P) := \forall P \xrightarrow{c} P'$ where $c \in \mathscr{H}$, it holds that $[\![P]\!] \xrightarrow{i(c)_l} [\![P']\!]$.
Trivial case: $\delta(\mathbf{0})$ is true.

- *Prefixing:* Suppose $S = c.P$, $\delta(P)$, then $[\![S]\!] = i(c)_l.[\![P]\!]$. Let $S' = P$ we have the proof.
- *Choice:* Straightforward.
- *Parallel composition:* Suppose $S = P|Q$, $S \xrightarrow{c} S'$, $\delta(P)$, $\delta(Q)$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$. Because $c \neq \tau$, the transition stems inevitably from *interleaving*. Suppose it is left interleaving, then we have $P \xrightarrow{c} P'$ and $S' = P'|Q$. Because of $\delta(P)$, and Lemma 3, we can compose a path:

  $$[\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{i(c)_l} [\![P']\!]|[\![Q]\!] = [\![S']\!]$$

  and we have the proof. In case of right interleaving, the proof is similar.

□

Lemma 5 *If P is an ABC process, for all $P \xrightarrow{b?} P'$ where $b? \in \mathscr{B}?$, $\exists n \in \mathbb{N}^+$ such that $[\![P]\!] \xrightarrow{i(b?)_h}{}^n \xrightarrow{\tau_m}{}^n [\![P']\!]$.*

Proof: We prove by induction.
Let the induction hypothesis be: $\Delta(P)$: the lemma is true for process P.
Trivial case: $\delta(\mathbf{0})$ is true.

- *Prefixing:* Suppose $S = b?.P$, $\Delta(P)$, then $[\![S]\!] = i(b?)_h.\tau_m.[\![P]\!]$, $n = 1$. Let $S' = P$ we have the proof.
- *Choice:* Straightforward.
- *Parallel composition:* Suppose $S = P|Q$, $S \xrightarrow{b?} S'$, $\Delta(P)$, $\Delta(Q)$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$.

– If the transition stems from interleaving, because of Lemma 3 we can easily compose a path that proves $\Delta(S)$.
– If the transition stems from a broadcast-synchronization, which means $P \xrightarrow{b?} P'$ and $Q \xrightarrow{b?} Q'$, $S' = P'|Q'$, and then $[\![P]\!] \xrightarrow{i(b?)_h}{}^{n} R \xrightarrow{\tau_m}{}^{n} [\![P']\!]$, $[\![Q]\!] \xrightarrow{i(b?)_h}{}^{j} T \xrightarrow{\tau_m}{}^{j} [\![Q']\!]$. $R,T \xrightarrow{\tau_m}$ implies $R,T \xrightarrow{\tau_h}\!\!\!\!\!\!/$. Because of Lemma 3, the fact that high prioritised actions cannot be preempt, and that $R,T \xrightarrow{\tau_h}\!\!\!\!\!\!/$, we can compose a path:

$$[\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{i(b?)_h}{}^{n+j} R|T \xrightarrow{\tau_m}{}^{n+j} [\![P']\!]|[\![Q']\!] = [\![S']\!]$$

and we have the proof.

□

Lemma 6 *$\forall j \in \mathbb{N}$, if $P \xrightarrow{b!} P'$, then $\exists n \in \mathbb{N}$ such that $[\![P]\!] \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n} \xrightarrow{\overline{i(b?)}_h}{}^{j} R \xrightarrow{\tau_m}{}^{n+1} [\![P']\!]$, and $R \xrightarrow{\tau_h}\!\!\!\!\!\!/$.*

Proof: We prove by induction. Let the induction hypothesis be $\theta(P)$: the lemma is true for process P. Trivial case: $\theta(\mathbf{0})$ is true.

- *Prefixing:* Suppose $S = b!.P$, $\theta(P)$, then $[\![S]\!] = \tau_l.(\overline{i(b?)}_h.X + \tau_m.[\![P]\!])$ where $X := i(b?)_h.X + \tau_m.[\![P]\!]$. Let $S' = P$ we have the proof.
- *Choice:* Straightforward.
- *Parallel composition:* Suppose $S = P|Q$, $S \xrightarrow{b!} S'$, $\theta(P)$, $\theta(Q)$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$.
  – Suppose the transition stems from interleaving. Because of Lemma 3 and $\theta(P)$, $\theta(Q)$, we can construct a path that proves $\theta(S)$.
  – Suppose the transition stems from a broadcast communication, which means $P \xrightarrow{b!} P'$ and $Q \xrightarrow{b?} Q'$ or $P \xrightarrow{b?} P'$ and $Q \xrightarrow{b!} Q'$. We can assume it to be the first case. Because $\theta(P)$ and Lemma 5 we have paths:

$$[\![P]\!] \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n} \xrightarrow{\overline{i(b?)}_h}{}^{j} R \xrightarrow{\tau_m}{}^{n+1} [\![P']\!]$$

$$[\![Q]\!] \xrightarrow{i(b?)_h}{}^{k} T \xrightarrow{\tau_m}{}^{k} [\![Q']\!]$$

  $\forall j \in \mathbb{N}$, for certain n, k, and $R,T \xrightarrow{\tau_h}\!\!\!\!\!\!/$. Therefore we can compose a path

$$[\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n+k} R|T \xrightarrow{\tau_m}{}^{n+1+k} [\![P']\!]|[\![Q']\!] = [\![S']\!]$$

  and we have the proof.

□

Now we prove that Lemma 1 is true by discussing different labels of the transition.

Proof: (of Lemma 1) $\forall P \xrightarrow{\alpha}_1 P'$, we prove $[\![P]\!] \Longrightarrow_2 \asymp_2 [\![P']\!]$ where $\asymp_2$ is a strong bisimulation, $\alpha = \tau$ or $\alpha \in \mathscr{B}!$.

- If $\alpha = \tau$, because of Lemma 4 we have $[\![P]\!] \xrightarrow{\tau_l} [\![P']\!]$, which proves $[\![P]\!] \Longrightarrow_2 \asymp_2 [\![P']\!]$.
- If $\alpha \in \mathscr{B}!$, let $j = 0$ in Lemma 6 we have $[\![P]\!] \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n} \xrightarrow{\tau_m}{}^{n+1} [\![P']\!]$ for a certain $n \in \mathbb{N}$, which proves $[\![P]\!] \Longrightarrow_2 \asymp_2 [\![P']\!]$.

□

With Lemma 1 and Lemma 2, the (COMPLETE) criterion is proven.

Corollary 1 *From the proof of Lemma 1 we can see, the* (COMPLETE) *criteria can be strengthened as:*

$$\forall S \Longmapsto_1 S', \textit{ it holds that } [\![S]\!] \Longmapsto_2 [\![S']\!]$$

which means, the $\asymp$ *relation is strengthened to identity relation.*

Now we proceed to prove the (SOUND) criteria. Trivially, we can prove that the encoding respects the (SOUND) property if $T = [\![S]\!]$, by setting $S = S'$, we have $S \Longmapsto_1 S$ and $[\![S]\!] \Longmapsto_2 \asymp_2 [\![S]\!]$. Then we only need to discuss the case where $T \neq [\![S]\!]$.

Proof: Induction hypothesis $\zeta(P)$: if $[\![P]\!] \Longmapsto_2 T$, then there exists P' such that $P \Longmapsto_1 P'$, $T \Longmapsto_2 \asymp_2 [\![P']\!]$. Trivial case: $\zeta(\mathbf{0})$ is true, for there is no transition from $\mathbf{0}$ to a state that is not $\mathbf{0}$: $[\![\mathbf{0}]\!] \not\longrightarrow$.

- *Prefixing:* Suppose $S = \alpha.P$, $\alpha \in Act_{ABC}$, $\zeta(P)$.

$$[\![S]\!] = [\![\alpha.P]\!] = \begin{cases} \tau_l.(\overline{i(b?)}_h.X + \tau_m.[\![P]\!]) \text{ where } X := i(b?)_h.X + \tau_m.[\![P]\!] & \text{if } \alpha = b! \in \mathscr{B}! \\ i(b?)_h.\tau_m.[\![P]\!] & \text{if } \alpha = b? \in \mathscr{B}? \\ i(\eta)_l.[\![P]\!] & \text{if } \alpha = \eta \in \mathscr{H} \cup \{\tau\} \end{cases}$$

  Because $i(b?) \neq \tau$, we can leave out the case where $\alpha = b?$, for the $\Longmapsto_2$ is defined as the reflexive, transitive closure of $\xrightarrow{\tau}$.

  As can be seen from the equation above, we can divide all the states on the path starting from $[\![S]\!]$ into two groups: after $[\![P]\!]$ (including $[\![P]\!]$), or not.

$$\textit{After(including)}[\![P]\!] : [\![S]\!] \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow [\![P]\!] \longrightarrow \cdots \longrightarrow T \longrightarrow \cdots$$

  - Once T is after(including) the execution of $[\![P]\!]$, because of $\zeta(P)$, we can find a P' such that $P \Longmapsto_1 P'$ and $T \Longmapsto_2 \asymp_2 [\![P']\!]$, then we can compose a path: $[\![S]\!] \Longmapsto_2 [\![P]\!] \Longmapsto_2 T$, such that $S \Longmapsto_1 P \Longmapsto_1 P'$, and $T \Longmapsto_2 \asymp_2 [\![P']\!]$. Let $S' = P'$, we have the proof for $\zeta(S)$.
  - If T is strictly before $[\![P]\!]$, then it must be $T = X$. Then we can execute an τ_m and reach the state $[\![P]\!]$, and prove like the case before.

- *Choice:* Suppose $S = P + Q$, $\zeta(P)$, $\zeta(Q)$, then $[\![S]\!] = [\![P]\!] + [\![Q]\!]$. If $[\![S]\!] \Longmapsto_2 T$ because $[\![P]\!] \Longmapsto_2 T$, then there exists P' such that $P \Longmapsto_1 P'$, $T \Longmapsto_2 \asymp_2 [\![P']\!]$ because of $\zeta(P)$. Let $S' = P'$, then we can compose a transition $[\![S]\!] = [\![P]\!] + [\![Q]\!] \Longmapsto_2 T$, such that

$$S = P + Q \Longmapsto_1 S' = P', T \Longmapsto_2 \asymp_2 [\![S']\!] = [\![P']\!]$$

  which proves $\zeta(S)$ true.

- *Parallel composition:* Suppose $S = P|Q$, $[\![S]\!] \Longmapsto_2 T$, $\zeta(P)$, $\zeta(Q)$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$. We decompose the path $[\![S]\!] \Longmapsto_2 T$ like in Section 3:

$$[\![P]\!] \longrightarrow\!\!\!\!\rightarrow \cdots \longrightarrow\!\!\!\!\rightarrow R$$

$$[\![Q]\!] \longrightarrow\!\!\!\!\rightarrow \cdots \longrightarrow\!\!\!\!\rightarrow S$$

  where $T = R|S$.

Because of $\zeta(P)$ and $\zeta(Q)$ we can find some P' and some Q' such that:

$$P \Longmapsto_1 P', R \Longmapsto_2 \asymp_2 [\![P']\!]$$

$$Q \Longmapsto_1 Q', S \Longmapsto_2 \asymp_2 [\![Q']\!]$$

Because of Lemma 7 below, we can find such P', Q' that $R|S \Longmapsto_2 \asymp [\![P']\!]|[\![Q']\!]$ and $P|Q \Longmapsto_1 P'|Q'$. Let $S' = P'|Q'$ we have the proof for $\zeta(S)$.

□

Lemma 7 *We can find such P', Q' that $R|S \Longmapsto_2 \asymp [\![P']\!]|[\![Q']\!]$ and $P|Q \Longmapsto_1 P'|Q'$.*

Proof: With the following two lemmas. □

Let R be the relation that relates a state in the source language with all its middle states of the transition.

Lemma 8 *If $S_1 \mathrm{R} T_1$ and $T_1 \xrightarrow{\alpha} T_2$ then $\exists S_2$ such that $S_1 \Longmapsto S_2$ and $S_2 \mathrm{R} T_2$.*

Proof: □

Lemma 9 *If $S_1 \mathrm{R} T_1$ then $\exists S_2, T_2$ such that $T_1 \Longmapsto T_2$ and $S_1 \Longmapsto T_2$ and $T_2 \asymp [\![S_2]\!]$.*

Proof: □

6.2.4 Divergence reflection

Property 4 (Divergence reflection) *A translation $[\![\cdot]\!]$: $L_1 \to L_2$ reflects divergence if, for every S such that $[\![S]\!] \longmapsto_2^w$, it holds that $S \longmapsto_1^w$, where*

$$S \longmapsto^w \text{ iff } \exists S \xrightarrow{\gamma_1} s_1 \xrightarrow{\gamma_2} s_2 \xrightarrow{\gamma_3} s_3 \xrightarrow{\gamma_4} \cdots \text{ (infinite)}$$

where γ_i is silent action or $\gamma_i \in \mathscr{B}!$, in case that S is a process in ABC. We call S divergent *if $S \longmapsto^w$.*

Proof: The property states, that the encoding does not introduce divergence that did not exist. We can see from the encoding, it maps an ABC process to a CCS_P process, only introducing infinite expression when translating $[\![b!.P]\!] = \tau_l.(\overline{i(b?)}_h.X + \tau_m.[\![P]\!])$ where $X = i(b?)_h.X + \tau_m.[\![P]\!]$. Then we can easily conclude: if $[\![S]\!] \longmapsto_2^w$, not involving $[\![b!]\!]$, then $S \longmapsto_1^w$.

If $[\![S]\!] \longmapsto_2^w$ where some process in the path is doing broadcast communication, resulting in some τ_h. Because of *Lemma* 5 and *Lemma* 6 we can see that the number of τ_h corresponds to the number of $b?$, and thus $[\![S]\!]$ is divergent only if S is divergent. □

6.2.5 Success sensitiveness

Property 5 (Success sensitiveness)

References

[1] VAN GLABBEEK, Rob; HÖFNER, Peter. Progress, fairness and justness in process algebra. arXiv preprint arXiv:1501.03268, 2015.

[2] VAN GLABBEEK, Rob. Component. In progress.

[3] CLEAVELAND, Rance; HENNESSY, Matthew. Priorities in process algebras. Information and Computation, Volume 87, Issues 1–2, 1990, Pages 58-77, ISSN 0890-5401.

[4] DE BORTOLI, Filippo. A Justness-aware Semantics Project Report. 2018.

[5] GORLA, Daniele. Towards a unified approach to encodability and separation results for process calculi, Information and Computation, Volume 208, Issue 9, 2010, Pages 1031-1053, ISSN 0890-5401.

[6] VERSARI, C., BUSI, N., GORRIERI, R. (2009). An expressiveness study of priority in process calculi. Mathematical Structures in Computer Science, 19(6), 1161-1189. doi:10.1017/S0960129509990168