Expressiveness of Process Algebras with Signals, Broadcast, and Priorities

Anran Wang

Contents

1 Introduction

Since its appearance in 1982, the *Calculus of Communicating Systems (CCS)* [10] has been studied, compared with other calculi, and equipped with more mechanisms. The resulting new algebras can model certain distributed systems in a more precise and straightforward fashion, but also raise a question: do they only change the written form of the models, or are they truly more expressive than pure CCS?

Expressiveness in this context means the ability of a language to model, represent, portray entities, systems, protocols, etc. If the expressiveness is described in a direct way, such as “CCS can model addition”, then it is called *absolute expressiveness*. If it is described comparatively, such as “A is more expressive than B”, then it is called *relative expressiveness*. To compare the expressiveness of different calculi, one usually takes two approaches: for positive results, one *encodes* one language into another. Once such an encoding is established, one can infer that the target language is at least as expressive as the source. Since each term in the source language can be encoded into a term in the target language, every model described in the source can therefore also be described in the target language. To decide how “good” the encoding is, one refers to *encodability criteria*, which describe how structured the encoding is, how many behaviours it preserves, etc. The criteria play an essential rule in expressiveness studies. An encoding can be treated as “good” due to some set of criteria, but fails to conform to another set that is either more strict or totally different.

January 7, 2021

For negative results, one searches for *separation results*. If one can find a system, algorithm, etc. that can be modelled by one algebra but not by another, one can conclude that this separates the two algebras, and the first algebra cannot be expressed by the other.

In this thesis, we first present at the beginning of Section 2 some relevant definitions. The syntax and semantics of the process algebras to be explored are explained in Sections 2.1, 2.2, 2.3, and 2.4, introducing CCS, CCS with Signals (CCSS), the Algebra of Broadcast Communication (ABC), CCS with Priorities (CCS^{sg}), respectively.

Expressiveness criteria consulted in this thesis are listed in Section 3.1. Encodings are given in Section 3.2 from CCS to CCSS/ABC/CCS^{sg} and vise versa. Separation results are exhibited in Section 3.3 as well as Section 3.4 concerning CCSS and ABC, respectively. We proceeded to an encoding from ABC to CCS^{sg} in Section 3.5. In the end, we separated CCS^{sg} from other calculi in Section 3.6.

Finally, a map of relative expressive power is shown in Section 4.

2 Preliminaries: CCS and its Extensions with Signals, Broadcasts, and Priorities

In general, the symbol $\mathscr{N}$ represents the set of all possible names. $\mathscr{K}$ is the set of all possible agent identifiers, and $\mathscr{T}$ is the set of all possible process terms.

When $\mathscr{N}$, $\mathscr{K}$, and $\mathscr{T}$ are equipped with indices, then the symbols indicate the subset of names, agent identifiers, and terms in the specific process algebra. For example, $\mathscr{N}_{CCS} \subseteq \mathscr{N}$ is the subset of names that parameterise CCS.

Definition 1 (LTS) A *labelled transition system* (LTS) is a tuple $(\mathscr{T}, \mathscr{L}, \longrightarrow)$ with $\mathscr{T}$ set of states, $\mathscr{L}$ set of labels. $\longrightarrow$ is a transition relation that is a subset of $\mathscr{T} \times \mathscr{L} \times \mathscr{T}$. The transition relation is usually written as *source* $\xrightarrow{label}$ *target*.

An LTS is *rooted* if there is a unique state s_0 called *root*.

Definition 2 (Reduction) The *reduction relation* is a subset of a given transition relation, where the set of labels includes only *reduction labels*.

A transition is called a *reduction* if it is labelled with an $l \in \mathscr{N}_r$ and is written *source* $\longrightarrow$ *target*.

Formally, $P \longrightarrow Q$ if and only if $\exists \alpha \in \mathscr{N}_r : \ P \xrightarrow{\alpha} Q$.

Usually, the set of reduction labels $\mathscr{N}_r$ only includes the silent action τ, such as in CCS. The set of reduction labels should only include labels of actions that can occur autonomously. For instance, the action "start" of an automobile waiting at a crossroad should not be labelled with a reduction label, for it depends on the traffic light doing "signal green"; the action "signal green", however, should be labelled with a label from $\mathscr{N}_r$, because it happens according to its own logic, not dependenting on its surroundings; it is indifferent however many cars are waiting.

Definition 3 (Path) A *path* in a labelled transition system L is a sequence of transitions, where the target state of one transition is the source state of the next transition. A path π is written in the form:

$$\pi : \ s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \cdots \xrightarrow{\alpha_n} s_n$$

for some $n \in \mathbb{N}$, or

$$\pi : \ s_0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \cdots$$

if the path is infinite. The *source* of a path is the first state in the path, noted as $source(\pi)$. We note "the path *of* P" if the source of the mentioned path is P. The states in the path satisfy $s_n \xrightarrow{\alpha_{n+1}} s_{n+1}$, for all relevant n. When $n = 0$ the path contains only one state.

A path is *maximal*, if it is either infinite or ends in a state with no outgoing transitions in L.

A path is a *computation* if all transitions are reductions. A computation is maximal, if it as a path is maximal.

Definition 4 (Barb) A state $P \in \mathscr{T}$ *exhibits barb* or *barbs* $x \in \mathscr{N} \uplus \overline{\mathscr{N}}$, written $P \downarrow x$, if and only if $P \xrightarrow{x}$, where $P \xrightarrow{x}$ if and only if $\exists P' \in \mathscr{T} : \ P \xrightarrow{x} P'$.

A path π exhibits barb x, written $\pi \downarrow x$, if and only if there exists at least one state s_i within π such that $s_i \downarrow x$. The collection of barbs forms the set of observables for π, written $Obs(\pi) = \{x \mid \pi \downarrow x\}$.

A state P *weakly barbs* x, written $P \Downarrow x$ if and only if $\exists P' : \ (P \Longrightarrow P') \wedge (P' \downarrow x)$, where $\Longrightarrow$ is the reflexive, transitive closure over the reduction relation $source \longrightarrow target$.

$P \not\downarrow x$ if and only if P does not barb x, $P \not\Downarrow x$ if and only if P does not weakly barb x.

Definition 5 (Process Graph) A *process* is an element of the set $\mathscr{T}$ of process terms. Given a process algebra PA, all elements in $\mathscr{T}_{PA}$ are called PA processes.

A *process graph* of a process P is a rooted labelled transition system $(\mathscr{T}, \mathscr{L}, \longrightarrow, P)$ where P is the root, $\mathscr{T}$ is the set of process terms, $\mathscr{L}$ the set of labels a.k.a. process names, $\longrightarrow$ the transition relation. $P \xrightarrow{\alpha} P'$ means process P can do an action α and reach state P', which is also a process.

2.1 CCS: Calculus of Communicating Systems

The Calculus of Communicating Systems (CCS) [10] is a process algebra parameterised with $\mathscr{K}_{CCS}$ and $\mathscr{N}_{CCS}$. The set of handshake actions is $\mathscr{H}_{CCS} := \mathscr{N}_{CCS} \uplus \overline{\mathscr{N}_{CCS}}$ where $\overline{\mathscr{N}_{CCS}} := \{\overline{c} \mid c \in \mathscr{N}_{CCS}\}$, $\uplus$ is the disjoint union[1], $\bar{\cdot}$ denotes the complement of a name, it satisfies $\overline{\overline{c}} = c$ for $c \in \mathscr{N}_{CCS}$ and $\overline{\tau} = \tau$, where τ is a special name not in $\mathscr{N}_{CCS}$ nor $\mathscr{N}$ that represents a silent action. In CCS, τ is the only reduction label.

The set of actions $\mathscr{A}_{CCS} := \mathscr{H}_{CCS} \uplus \{\tau\}$.

In the remaining subsection, let c range over $\mathscr{H}_{CCS}$, α over $\mathscr{A}_{CCS}$, A over $\mathscr{K}_{CCS}$. The class $\mathscr{T}_{CCS}$ of CCS expressions is the smallest set including:

Table 1: CCS Expressions

• *prefixes* $\alpha.P$	• *(infinite) choices* $\sum_{i \in I} P_i$
• *restrictions* $P \backslash L$	• *parallel compositions* $P \mid Q$
• *relabellings* $P[f]$	• *agent identifiers* A

where $P, Q \in \mathscr{T}_{CCS}$, $A \in \mathscr{K}_{CCS}$, I is an index set, $L \subseteq \mathscr{N}_{CCS}$ a subset of names, $f : \mathscr{H}_{CCS} \rightarrow \mathscr{H}_{CCS}$ a renaming function satisfying $f(\overline{a}) = \overline{f(a)}$ and extends to $\mathscr{A}_{CCS}$ by defining $f(\tau) = \tau$. $\mathbf{0}$ (nil) represents inaction, defined as $\mathbf{0} := \sum_{i \in \emptyset} P_i$. In case $i \in \{1, 2\}$, we write $P_1 + P_2$ instead of $\sum_{i \in \{1,2\}} P_i$ for readability.

In CCS as well as its extentions, each agent identifier comes with a defining equation $A \overset{def}{=} P$ where P is a CCS term and does not contain occurrances of *unguarded* agent identifiers. An agent identifier A is guarded, if there is $\alpha \in \mathscr{A}_{CCS}$ as prefix of $A : \alpha.A$. Otherwise it is unguarded. The same is required for agent identifiers in CCSS, ABC, and CCS^{sg}.

[1]The disjoint union $\uplus$ implies that the sets it joins are disjoint.

Table 2: Structural Operational Semantics of CCS

$$\frac{}{\alpha.P \xrightarrow{\alpha} P}\ (\text{ACT}) \qquad \frac{P_j \xrightarrow{\alpha} P',\ j \in I}{\sum_{i\in I} P_i \xrightarrow{\alpha} P'}\ (\text{SUM})$$

$$\frac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\ (\text{PAR-L}) \qquad \frac{Q \xrightarrow{\alpha} Q'}{P|Q \xrightarrow{\alpha} P|Q'}\ (\text{PAR-R}) \qquad \frac{P \xrightarrow{c} P',\ Q \xrightarrow{\bar{c}} Q'}{P|Q \xrightarrow{\tau} P'|Q'}\ (\text{COMM})$$

$$\frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}\ (\text{REL}) \qquad \frac{P \xrightarrow{\alpha} P', \alpha, \bar{\alpha} \notin L}{P\backslash L \xrightarrow{\alpha} P'\backslash L}\ (\text{RES}) \qquad \frac{P \xrightarrow{\alpha} P'}{A \xrightarrow{\alpha} P'}(A \overset{def}{=} P)\ (\text{REC})$$

The symbol $\overset{def}{=}$ is in principle the same as $:=$, we use it to emphasise the (possible) appearance of recursions. Recursions are defined using agent identifiers, for example $P \overset{def}{=} \alpha.P$ is a process that does action α repeatedly.

Let $\mathscr{T} = \mathscr{T}_{CCS}$, $\mathscr{L} = \mathscr{A}_{CCS}$ in Definition 1, CCS yields a labelled transition system. Based on LTS, the semantics of an algebra can be given by means of *structural operational semantics (SOS)*. A rule in an SOS table takes the following form:

$$\frac{\text{set of condition(s)}}{\text{set of result(s)}}$$

This rule means "if the conjunction of the conditions is true, then the results can happen". When the set of conditions in an SOS rule is empty, this rule is called an *axiom*, and the results can always happen.

All CCS transitions follow the rules in Table 2.

2.2 CCS with Signals

CCS with Signals (CCSS) [4] is acquired by extending CCS with a set of *signals* as well as a *siganlling* operator. The set of names in CCSS is defined as $\mathscr{N}_{CCSS} := \mathscr{N}_{CCS} \uplus \mathscr{S}$ where $\mathscr{S}$ is the set of signals. Note that the only names in $\mathscr{N}_{CCS}$ have complement.

The set of handshake actions is $\mathscr{H}_{CCSS} := \mathscr{H}_{CCS}$. The set of actions in CCSS is defined by $\mathscr{A}_{CCSS} := \mathscr{S} \uplus \mathscr{H}_{CCSS} \uplus \{\tau\}$. The set of agent identifiers $\mathscr{K}_{CCSS} := \mathscr{K}_{CCS}$.

The binary operator signalling binds a CCSS process P and a signal s in such form: $P\hat{\ }s$.

The class $\mathscr{T}_{CCSS}$ of CCSS expressions is the smallest set including:

Table 3: CCSS Expressions

• *prefixes* $\alpha.P$	• *(infinite) choices* $\sum_{i\in I} P_i$	• *signalling* $P\hat{\ }s$
• *restrictions* $P\backslash L$	• *parallel compositions* $P\vert Q$	
• *relabellings* $P[f]$	• *agent identifiers* A	

where $\alpha \in \mathscr{A}_{CCSS}$, I an index, $P, Q \in \mathscr{T}_{CCSS}$, $A \in \mathscr{K}_{CCSS}$, $s \in \mathscr{S}$, $L \subseteq \mathscr{N}_{CCSS}$, $f : (\mathscr{S} \to \mathscr{S}) \cup (\mathscr{H}_{CCSS} \to \mathscr{H}_{CCSS})$ satisfying $f(\overline{a}) = \overline{f(a)}$, it extends to $\mathscr{A}_{CCSS}$ by defining $f(\tau) = \tau$.

The semantics of CCSS is extended from Table 2 with Table 4. The curved arrow symbol $\curvearrowright$ indicates the ability of a CCSS process to emit signals. The collection of reduction labels in CCSS is $\{\tau\}$.

Table 4: Structural Operational Semantics of Signals in CCSS

$$(P\hat{\ }s)^{\frown s} \qquad \frac{P \xrightarrow{\alpha} P'}{P\hat{\ }r \xrightarrow{\alpha} P'} \qquad \frac{P_i^{\frown s},\ i \in I}{\sum_{i\in I} P_i^{\frown s}}$$

$$\frac{P^{\frown s}}{(P|Q)^{\frown s}} \qquad \frac{P^{\frown s},\ Q \xrightarrow{s} Q'}{P|Q \xrightarrow{\tau} P|Q'} \qquad \frac{P \xrightarrow{s} P',\ Q^{\frown s}}{P|Q \xrightarrow{\tau} P'|Q} \qquad \frac{Q^{\frown s}}{(P|Q)^{\frown s}}$$

$$\frac{P^{\frown s}}{(P\hat{\ }r)^{\frown s}} \qquad \frac{P^{\frown s}}{(P\backslash L)^{\frown s}}\ (s \notin L) \qquad \frac{P^{\frown s}}{P[f]^{\frown f(s)}} \qquad \frac{P^{\frown s}}{A^{\frown s}}\ (A \overset{def}{=} P)$$

Table 5: Structural Operational Semantics of ABC

$$\frac{P \xrightarrow{b\sharp_1} P',\ Q \overset{b?}{\not\longrightarrow}}{P|Q \xrightarrow{b\sharp_1} P'|Q}\ (\textsc{Bro-l}) \qquad \frac{P \xrightarrow{b\sharp_1} P',\ Q \xrightarrow{b\sharp_2} Q'}{P|Q \xrightarrow{b\sharp} P'|Q'}\ (\textsc{Bro-c})$$

$$\frac{P \overset{b?}{\not\longrightarrow},\ Q \xrightarrow{b\sharp_2} Q'}{P|Q \xrightarrow{b\sharp_2} P|Q'}\ (\textsc{Bro-r}) \qquad \sharp_1 \circ \sharp_2 = \sharp \neq \_ \ \text{ with } \begin{array}{c|cc} \circ & ! & ? \\ \hline ! & \_ & ! \\ ? & ! & ? \end{array}$$

2.3 Algebra of Broadcast Communication

The Algebra of Broadcast Communication (ABC) [7] is an extension of CCS, combined with features from the Calculus of Broadcasting Systems (CBS) [13].

The set of names in ABC is extended with broadcast names $\mathscr{B}$, they are sometimes called *channels*. The sets of *broadcast* and *receive* actions are given by $\mathscr{B}\sharp = \{b\sharp \mid b \in \mathscr{B}\}$ where $\sharp \in \{!, ?\}$. The exclamation mark ! is the notation for broadcasting and the question mark ? for receiving. As a result, $\mathscr{N}_{ABC} := \mathscr{N}_{CCS} \uplus \mathscr{B}$, $\mathscr{A}_{ABC} := \mathscr{H}_{ABC} \uplus \mathscr{B}! \uplus \mathscr{B}? \uplus \{\tau\}$ where $\mathscr{H}_{ABC} := \mathscr{H}_{CCS}$. A relabelling function is $f : (\mathscr{B} \to \mathscr{B}) \cup (\mathscr{N}_{CCS} \to \mathscr{N}_{CCS})$, it extends to $\mathscr{A}_{ABC}$ by $f(\bar{c}) = \overline{f(c)}$, $f(b\sharp) = f(b)\sharp$, $f(\tau) = \tau$. The set of agent identifiers in ABC $\mathscr{K}_{ABC} := \mathscr{K}_{CCS}$.

Note that only names in $\mathscr{B}$ have complement.

The class of ABC terms is defined by Table 1, where $\alpha \in \mathscr{A}_{ABC}$, $L \subseteq \mathscr{H}_{ABC}$[2], f a relabelling function, $P, Q \in \mathscr{T}_{ABC}$, and $A \in \mathscr{K}_{ABC}$.

The semantics of ABC is extended from Table 2 with rules in Table 5, where $\sharp_1$, $\sharp_2$, $\sharp \in \{!, ?\}$.

The reduction relation in ABC is defined as $\{\tau\} \cup \mathscr{B}!$, considering broadcast actions happen autonomously, and should not be blocked in any way.

2.4 CCS with Priorities

Cleaveland, Lüttgen, and Natarajan categorised process-algebraic priorities by combining its two characteristics: *static* or *dynamic priorities*, and *local* or *global preemption* [3]. In this thesis, we are going to look into CCS with static priority and global preemption, noted as *CCS*^{sg} [3].

The names in CCS^{sg} always include a priority level. A prioritised name is of the form $a{:}k$ or $\overline{a{:}k}$ where $k \in \mathbb{N}$ is a priority level, with 0 being the highest. The set of CCS^{sg} names is $\mathscr{N}_{CCS^{sg}} := \{a{:}k \mid$

[2]This implies that neither silent nor broadcast actions can be restricted.

Table 6: Potential Initial Action Sets for CCS^{sg}

$I(\alpha{:}k.P) := \{\alpha{:}k\}$	$I(P[f]) := \{f(\alpha{:}k) \mid \alpha{:}k \in I(P)\}$
$I(P\backslash L_p) := I(P)\backslash(L_p \cup \overline{L_p})$	$I(\sum_{i\in I} P_i) := \bigcup_{i\in I} I(P_i)$ (I an index set)
$I(A) := I(P)$ ($A \overset{def}{=} P$)	$I(P\|Q) := I(P) \cup I(Q) \cup \{\tau{:}k \mid I(P) \cap \overline{I(Q)} \neq \emptyset\}$

$a \in \mathcal{N}_{CCS},\ k \in \mathbb{N}\}$, $\overline{\mathcal{N}_{CCS^{sg}}} := \{\overline{a{:}k} \mid a{:}k \in \mathcal{N}_{CCS^{sg}}\}$. Additionally, the complementat of a name satisfies $\overline{\alpha{:}k} = \overline{\alpha}{:}k$. The set of agent identifiers $\mathcal{K}_{CCS^{sg}} := \mathcal{K}_{CCS}$.

Any relabelling function $f : \mathcal{N}_{CCS} \to \mathcal{N}_{CCS}$ extends to CCS^{sg} names by defining $f(\alpha{:}k) := f(\alpha){:}k$ for $\alpha \in \mathcal{N}_{CCS}$ and $k \in \mathbb{N}$ a priority level. Note that relabelling functions do not change priority levels.

The restricted set $L_p \subseteq \mathcal{N}_{CCS^{sg}}$ and $\overline{L_p} := \{\overline{\alpha{:}k} \mid \alpha{:}k \in L_p\}$.

The sets of handshake actions and actions are defined in CCS^{sg} similarly as in CCS, $\mathcal{H}_{CCS^{sg}} := \mathcal{N}_{CCS^{sg}} \uplus \overline{\mathcal{N}_{CCS^{sg}}}$, $\mathcal{A}_{CCS^{sg}} := \mathcal{H}_{CCS^{sg}} \uplus \{\tau{:}k \mid k \in \mathbb{N}\}$. The set of reduction relations is the set containing silent actions with all priorities: $\{\tau{:}k \mid k \in \mathbb{N}\}$.

Note that co-names have the same priorities , i.e. the co-name of $a{:}0$ and the co-name of $a{:}1$ are different. In fact, actions with different priorities do not communicate.

To illustrate the operational semantics of CCS^{sg}, we need to first build *potential initial action sets* I, as shown in Table 6 ($k \in \mathbb{N}$). Intuitively, the potential initial action sets include actions that a process can perform according to SOS rules of CCS. At this stage we do not consider the possibility that some actions may be preempted (Definition 6) by others, thus the actions are *potential*. The potential initial action set of a process is built merely upon the structure of the process, thus *initial*.

In Table 6, the set $\overline{I(P)}$ is defined as $\overline{I(P)} := \{\overline{a{:}k} \mid a{:}k \in I(P)\}$, $P\backslash L_P$ is restriction in CCS^{sg}, and $I(P)\backslash(L_p \cup \overline{L_p})$ is the usual set substraction.

When an action has the highest priority, its operational semantics coincide with Table 2, but we still include it in the left side of Table 7 for readability. When an action has priority $0 < l$, it can only happen when not preempted (Definition 6) by a higher prioritised $\tau{:}k$, as represented in the right side of Table 7 ($0 < l, k < l$, I is an index set, $\alpha{:}0$, $\alpha{:}l \in \mathcal{A}_{CCS^{sg}}$, $c, \overline{c} \in \mathcal{H}_{CCS^{sg}}$).

Definition 6 A process P preempts the execution of $\alpha{:}l$, if and only if $P \xrightarrow{\tau:k} \wedge\ k < l$.

So far we have established different sets of names, Figure 1 helps clear the relationship between the sets.

The structural congruence relation $\equiv\ \in \mathcal{T}_{AG} \times \mathcal{T}_{AG}$, where $AG \in \{$CCS, CCSS, ABC, CCS$^{sg}\}$, is the smallest relation defined with the axioms in Table 8, for all $x \in \mathcal{T}_{AG}$. Also, we write $x.\mathbf{0}$ as x for short.

3 Expressiveness of CCS, CCSS, ABC, and CCS^{sg}

3.1 Expressiveness Criteria

Now we will explore the relative expressiveness of the aforementioned process algebras by constructing encodings or finding separation results. We establish some expressiveness criteria in this subsection, but readers are encouraged to read Definition 7 then skip the following pages to Section 3.2 and refer back when the encodability criteria are consulted.

In his survey [11], Parrow gave an overview on approaches to explore expressiveness, as well as many feasible criteria, some of which are applicable for the purpose of this thesis. Gorla summarised

Table 7: Structural Operational Semantics of CCS^{sg}

$\overline{\alpha{:}0.P \xrightarrow{\alpha:0} P}$	$\overline{\alpha{:}l.P \xrightarrow{\alpha:l} P}$
$\dfrac{P_j \xrightarrow{\alpha:0} P',\ j \in I}{\sum_{i\in I} P_i \xrightarrow{\alpha:0} P'}$	$\dfrac{P_j \xrightarrow{\alpha:l} P',\ j \in I, \forall i \neq j\ \forall k < l : \tau{:}k \notin I(P_i)}{\sum_{i\in I} P_i \xrightarrow{\alpha:l} P'}$
$\dfrac{P \xrightarrow{\alpha:0} P'}{P \mid Q \xrightarrow{\alpha:0} P' \mid Q}$	$\dfrac{P \xrightarrow{\alpha:l} P', \forall k < l : \tau{:}k \notin I(P \mid Q)}{P \mid Q \xrightarrow{\alpha:l} P' \mid Q}$
$\dfrac{Q \xrightarrow{\alpha:0} Q'}{P \mid Q \xrightarrow{\alpha:0} P \mid Q'}$	$\dfrac{Q \xrightarrow{\alpha:l} Q', \forall k < l : \tau{:}k \notin I(P \mid Q)}{P \mid Q \xrightarrow{\alpha:l} P' \mid Q}$
$\dfrac{P \xrightarrow{c:0} P',\ Q \xrightarrow{\overline{c:0}} Q'}{P \mid Q \xrightarrow{\tau:0} P' \mid Q'}$	$\dfrac{P \xrightarrow{c:l} P',\ Q \xrightarrow{\overline{c:l}} Q', \forall k < l : \tau{:}k \notin I(P \mid Q)}{P \mid Q \xrightarrow{\tau:l} P' \mid Q'}$
$\dfrac{P \xrightarrow{\alpha:0} P'}{P[f] \xrightarrow{f(\alpha:0)} P'[f]}$	$\dfrac{P \xrightarrow{\alpha:l} P'}{P[f] \xrightarrow{f(\alpha:l)} P'[f]}$
$\dfrac{P \xrightarrow{\alpha:0} P', \alpha{:}0, \overline{\alpha{:}0} \notin L_p}{P \backslash L_p \xrightarrow{\alpha:0} P' \backslash L_p}$	$\dfrac{P \xrightarrow{\alpha:l} P', \alpha{:}l, \overline{\alpha{:}l} \notin L_p}{P \backslash L_p \xrightarrow{\alpha:l} P' \backslash L_p}$
$\dfrac{P \xrightarrow{\alpha:0} P'}{A \xrightarrow{\alpha:0} P'} (A \overset{def}{=} P)$	$\dfrac{P \xrightarrow{\alpha:l} P'}{A \xrightarrow{\alpha:l} P'} (A \overset{def}{=} P)$

Table 8: Syntactic Equality Relation in CCS, CCSS, ABC, CCS^{sg}

$x + \mathbf{0} \equiv x$	$x \mid \mathbf{0} \equiv x$	
$x + x \equiv x$	$x + y \equiv y + x$	$x \mid y \equiv y \mid x$

some widely accepted criteria in [8], part of them are also used for this thesis. Also, the criteria used by Ene and Muntean [5], Versari, Busi and Gorrieri [14], Phillips [12] are applied here.

Definition 7 (Encoding) An *encoding* is a pair $(\llbracket\ \rrbracket, \varphi_{\llbracket\rrbracket})$ where $\llbracket\ \rrbracket$ is a mapping $\mathscr{T}_1 \to \mathscr{T}_2$ from the set of process expressions (process terms) $\mathscr{T}_1$ in the source language to the set of process terms $\mathscr{T}_2$ of the target language. $\varphi_{\llbracket\rrbracket}$ is a *renaming policy* $\mathscr{N} \to \mathscr{N}^k$ such that when $u \neq v$, $\varphi_{\llbracket\rrbracket}(u) = (u_1, u_2, \ldots, u_k)$, $\varphi_{\llbracket\rrbracket}(v) = (v_1, v_2, \ldots, v_k)$: $\forall i, j \in \{1, 2, \ldots, k\}, \nexists u_i = v_j$.

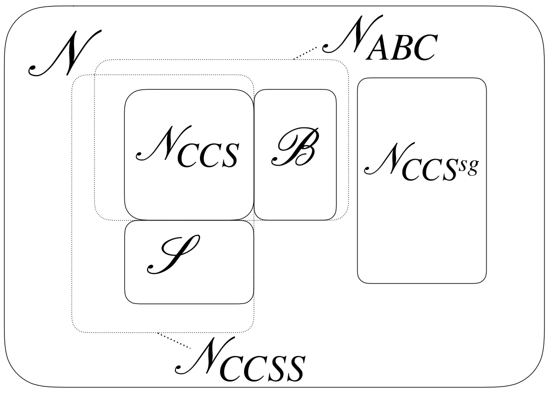
In most cases, a name in the source language is simply translated to itself; in some cases, a name is translated to another name. Either way, we omit the situation where a name is translated to a tuple of names and define the renaming policy as $\varphi_{\llbracket\rrbracket} : \mathscr{N} \to \mathscr{N}$.

Since there is now only one element in both $\varphi_{\llbracket\rrbracket}(u)$ and $\varphi_{\llbracket\rrbracket}(v)$, $\varphi_{\llbracket\rrbracket}(u)$ and $\varphi_{\llbracket\rrbracket}(v)$ share no common element (when $\varphi_{\llbracket\rrbracket}(u) \neq \varphi_{\llbracket\rrbracket}(v)$) implies that $\varphi_{\llbracket\rrbracket}()$ is injective.

Criterion 1 An encoding is *compositional*, if for each k-ary operator g in the source language, there exists a k-ary context $C_k(p_1, \ldots, p_k)$ in the target language such that $\llbracket g(s_1, \ldots, s_k) \rrbracket = C_k(\llbracket s_1 \rrbracket, \ldots, \llbracket s_k \rrbracket)$, where $p_1, \ldots, p_k$ stand for “holes” in the context to be filled with a concrete process.

Criterion 2 An encoding is *homomorphic* with respect to an operator g, if there is an identical operator in the target language, so that $\llbracket g(s_1, \ldots, s_k) \rrbracket = g(\llbracket s_1 \rrbracket, \ldots, \llbracket s_k \rrbracket)$.

Figure 1: Names in CCS as well as its Extensions



Homomorphy is a special case of compositionality.

Criterion 3 [5, 8] An encoding is *operational correspondent* up to a relation $\asymp_2$ if it satisfies:

1. (COMPLETE) if $S \Longrightarrow_1 S'$ then $[\![S]\!] \Longrightarrow_2 \asymp_2 [\![S']\!]$; and
2. (SOUND) if $[\![S]\!] \Longrightarrow_2 \asymp_2 [\![T]\!]$ then $S \Longrightarrow_1 S' \wedge [\![S']\!] \asymp_2 [\![T]\!]$ for some S'[3].

$\asymp_i \subseteq \mathscr{T}_i \times \mathscr{T}_i$ is an equivalence relation, $\Longrightarrow_i$ (Definition 4) is defined as the reflexive, transitive closure over the reduction relation $\longrightarrow_i$. $P \Longrightarrow_i \asymp_i Q$ if and only if $\exists P' \in \mathscr{T}_i : P' \asymp_i Q$.

The index indicates relations in the source or target language. We omit the use of indices if there is no confusion.

Criterion 4 [8] An encoding with $[\![\;]\!]$: $\mathscr{T}_1 \to \mathscr{T}_2$ *reflects divergence* if and only if

$$\forall S \in \mathscr{T}_1 : [\![S]\!] \longmapsto_2^{\omega} \text{ implies } S \longmapsto_1^{\omega} .$$

$P \longmapsto_i^{\omega}$, $i \in \{1,2\}$, if and only if there exists an infinite computation that has P as its source, and we say P is *divergent*.

Lemma 1 If the encoding $([\![\;]\!], \varphi_{[\![\,]\!]})$ reflects divergence, and the process $P \in \mathscr{T}_1$ is not divergent, then $[\![P]\!]$ is also not divergent.

Proof: Directly from Criterion 4. □

Criterion 5 [12, 14] Given a set of observables (Definition 4) $Obs \subseteq \mathscr{N} \cup \overline{\mathscr{N}}$, an encoding with $[\![\;]\!]$: $\mathscr{T}_1 \to \mathscr{T}_2$ is:

1. *observation-respecting* if $\forall P \in \mathscr{T}_1$:
   - for every maximal computation (Definition 3) $\mathscr{C}$ of P there exists a maximal computation $\mathscr{C}'$ of $[\![P]\!]$ such that $Obs(\mathscr{C}) = Obs(\mathscr{C}')$ (Definition 4); and
   - for every maximal computation $\mathscr{C}'$ of $[\![P]\!]$ there exists a maximal computation $\mathscr{C}$ of P such that $Obs(\mathscr{C}') = Obs(\mathscr{C})$.

[3]This criterion is incomparable with Gorla's criterion named "sound", and weaker than the criteria used by Ene and Muntean for "uniform encoding".

2. *weakly observation-respecting* if $\forall P \in \mathscr{T}_1$:
   - for every maximal computation $\mathscr{C}$ of P there exists a maximal computation $\mathscr{C}'$ of $[\![P]\!]$ such that $Obs(\mathscr{C}) = Obs(\mathscr{C}')$; and
   - for every maximal computation $\mathscr{C}'$ of $[\![P]\!]$ there exists a maximal computation $\mathscr{C}$ of P such that $Obs(\mathscr{C}') \subseteq Obs(\mathscr{C})$.

Informally, when an encoding is observation-respecting, any encoded term exhibits exactly so many observables as its source term; when an encoding is weakly observation-respecting, the encoded term are not allowed to exhibit more observales as its source term.

Criterion 6 [11] An encoding *preserves semantics* w.r.t. a behavioural equivalence $\asymp \subseteq ((\mathscr{T}_1 \cup \mathscr{T}_2) \times (\mathscr{T}_1 \cup \mathscr{T}_2))$, if the encoded term satisfies:

$$\forall P \in \mathscr{T}_1 : [\![P]\!] \asymp P.$$

This criterion is applicable, when the source and target algebras are similar, and there are equivalence relations defined over the union of the sets of source and target terms.

Strong bisimulation can be considered as one such equivalence relation.

Definition 8 (Strong Bisimulation) Let $P, Q \in \mathscr{T}$ be two processes, α is an action. P, Q are *strongly bisimilar* iff there is exists a binary relation $\mathscr{R}$ over $\mathscr{T}$ such that

(i) if $P\mathscr{R}Q$ then $\forall P \xrightarrow{\alpha} P' : \exists Q'$ s.t. $Q \xrightarrow{\alpha} Q' \wedge P'\mathscr{R}Q'$;

(ii) if $P\mathscr{R}Q$ then $\forall Q \xrightarrow{\alpha} Q' : \exists P'$ s.t. $P \xrightarrow{\alpha} P' \wedge P'\mathscr{R}Q'$;

The relation $\mathscr{R}$ is called a *strong bisimulation relation*.

A superset of the set of strong bisimulation relations is the set of *weak bisimulation* relations.

Definition 9 (Weak Bisimulation) Let $P, Q \in \mathscr{T}$ be two processes, α is an action. P, Q are *weakly bisimilar* iff there is exists a binary relation $\mathscr{R}$ over $\mathscr{T}$ such that

(i) if $P\mathscr{R}Q$ then $\forall P \xrightarrow{\alpha} P' : \exists Q'$ s.t. $Q \overset{\hat{\alpha}}{\Longrightarrow} Q' \wedge P'\mathscr{R}Q'$;

(ii) if $P\mathscr{R}Q$ then $\forall Q \xrightarrow{\alpha} Q' : \exists P'$ s.t. $P \overset{\hat{\alpha}}{\Longrightarrow} P' \wedge P'\mathscr{R}Q'$;

where $\overset{\hat{\alpha}}{\Longrightarrow} := \begin{cases} \Longrightarrow\xrightarrow{\alpha}\Longrightarrow, & \text{, if } \hat{\alpha} \neq \hat{\tau} \\ \Longrightarrow & \text{, else} \end{cases}$

The relation $\mathscr{R}$ is called a *weak bisimulation relation*.

Strong bisimulation relations are *finer* than weak bisimulation relations.

Definition 10 An equivalence relation $\asymp_a \subseteq \mathscr{T} \times \mathscr{T}$ is *finer* than $\asymp_b \subseteq \mathscr{T} \times \mathscr{T}$, if and only if for all $P, Q \in \mathscr{T}$, $P \asymp_a Q$ implies $P \asymp_b Q$, and there exists P', $Q' \in \mathscr{T} : P' \asymp_b Q' \wedge P' \not\asymp_a Q'$.

Definition 11 (Congruent) An equivalence relation $\asymp \in \mathscr{T} \times \mathscr{T}$ is *congruent* to an k-ary operator $g : \mathscr{T}^k \to \mathscr{T}$ if

$$\forall P_1, \ldots, P_k, Q_1, \ldots, Q_k \in \mathscr{T}, \forall 1 \leq i \leq k : P_i \asymp Q_i \text{ implies } g(P_1, \ldots, P_k) \asymp g(Q_1, \ldots, Q_k)$$

Definition 12 (Barb Preservation) Given a set of observables $Obs \subseteq \mathscr{N} \uplus \overline{\mathscr{N}}$, an equivalence relation $\asymp$ *preserves barb* if it satisfies:

$$\text{if } P \asymp Q \text{ then } (\forall x \in Obs : P \downarrow x \text{ iff } Q \downarrow x).$$

Definition 13 (Weak Barb Preservation) Given a set of observables $Obs \subseteq \mathcal{N} \uplus \overline{\mathcal{N}}$, an equivalence relation $\asymp$ *weakly preserves barb* if it satisfies:

$$\text{if } P \asymp Q \text{ then } (\forall x \in Obs : P \Downarrow x \text{ iff } Q \Downarrow x).$$

Criterion 7 An encoding from $\mathcal{T}_1$ to $\mathcal{T}_2$ *preserves barbs* if for a given set of barbs $Obs \subseteq \mathcal{N} \uplus \overline{\mathcal{N}}$:

$$\forall P \in \mathcal{T}_1, \forall x \in Obs : [\![P]\!] \downarrow_{\varphi_{[\![]\!]}(x)} \text{ if and only if } P \downarrow x.$$

Criterion 8 An encoding from $\mathcal{T}_1$ to $\mathcal{T}_2$ *weakly preserves barbs* if for a given set of barbs $Obs \subseteq \mathcal{N} \uplus \overline{\mathcal{N}}$:

$$\forall P \in \mathcal{T}_1, \forall x \in Obs : [\![P]\!] \Downarrow_{\varphi_{[\![]\!]}(x)} \text{ if and only if } P \Downarrow x.$$

3.2 CCS to its Extensions

We start by exploring the expressieness relation between CCS and CCSS, then extend the result to ABC and CCS^{sg}. First we encode CCS into CCSS. With CCSS being an extension of CCS, we can easily see that $\mathcal{T}_{CCS} \subseteq \mathcal{T}_{CCSS}$.

Proposition 1 There is an encoding from CCS to CCSS that respects all criteria mentioned in this thesis up to all reasonable equivalences.

Proof: The identity function with identity function as renaming policy is one such encoding:

$$\forall P \in \mathcal{T}_{CCS},\ \forall n \in \mathcal{N}_{CCS} :\ [\![P]\!] = P,\ \varphi_{[\![]\!]}(n) = n.$$

It is homomorphic, thus also compositional. It respects operational correspondence up to weak bisimulation, strong bisimulation, and it preserves semantics up to these equivalences. □

Proposition 2 There is an encoding from CCS to ABC and CCS^{sg} that respects all criteria mentioned in this thesis up to all reasonable equivalences.

Proof: Replacing CCSS with ABC in the encoding from CCS to CCSS, we have the encoding to ABC.

To encode CCS into CCS^{sg}, we simply rename all $n \in \mathcal{N}_{CCS}$ as $n{:}0 \in \mathcal{N}_{CCSsg}$.

This encoding is homomorphic, compositional, respects operational correspondence up to weak bisimulation, strong bisimulation, and it preserves semantics up to these equivalences. □

As a result, the extensions of CCS are at least as expressive as CCS, and the presented encodings respect a set of (strong) criteria. Now we present an encoding from the extensions to CCS.

Proposition 3 There exists a (non-compositional) encoding from CCSS to CCS that is divergence reflecting (Criterion 4), (weakly) observation-respecting (Criterion 5.2), semantics preserving (Criterion 6) and operational correspondent (Criterion 3) up to strong and weak bisimulation (Definitions 8, 9) as well as (weak) barb preservation (Definitions 12, 13).

Proof: In [6], van Glabbeek showed that CCS with infinite choices can express up to κ-*bounded* process graphs, where κ is an upper bound on the size of the index sets I allowed in summation $\sum_{i \in I}$. A set is κ-bounded if its cardinality is less than κ, a process graph is κ-bounded if for each its state there are less than κ outgoing transitions.

We show here that progress graphs of all CCSS processes are also κ-bounded.

First we need to relate CCSS to an LTS. The special operator ˆ and its semantics $\curvearrowright$ is not included in the typical labelled transition systems, we need to encode it so that every CCSS process term has a corresponding process graph.

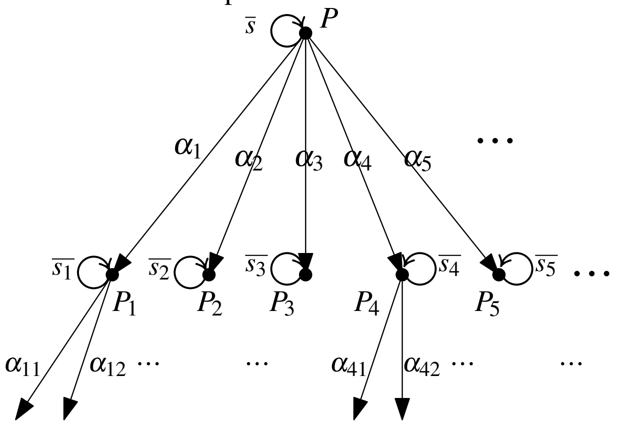
Bouwman gave an encoding that translates each process with signalling operator to a CCSS process without signalling operator [2]. As a result, the process graph of $P\hat{}s$ is the same as adding a loop from P to itself on the process graph of P.

Due to the infinite choice in CCSS, there are CCSS processes with infinitely many outgoing transitions. However, the infinite choice in CCSS is not different from that in CCS and there are still only maximal κ outgoing transitions.

As shown in Figure 2, the process graph of a CCSS process P is in general in such form: it has P as root, possibly with a loop from P to itself; there are (other than the loop) i (i less than κ) outgoing transitions, labelled with α_i, respectively; the target of each transition is noted P_i, where each P_i has possibly a loop to itself and j_i outgoing transitions (j_i less than κ).

Altogether, there are still less than κ outgoing transitions from states in the process graph of P, for any CCSS process P.

Figure 2: A General Representation of CCSS Process Graphs



In conclusion, the process graphs of CCSS processes are also κ-bounded and can be expressed by CCS terms. To find such expressions, we first present the process graph of a certain CCSS process, then choose one of the CCS processes that expresses this process graph, this CCS process then also expresses the original CCSS process.

This means that there exists an encoding from CCSS to CCS which satisfies all reasonable criteria (as mentioned in the proposition), but it is not compositional, as suggested by Proposition 5. □

Proposition 4 The result in Proposition 3 extends to all process algebras that generate up to κ-bounded process graphs, including ABC and CCS^{sg}.

Proof: Similar as the proof for Proposition 3. □

3.3 CCSS to CCS, ABC, and CCS^{sg}

Proposition 5 There is no compositional (Criterion 1), divergence-reflecting (Criterion 4) encoding from CCSS to CCS that preserves semantics (Criterion 6) up to weak bisimulation (Definition 9) or a finer (Definition 10) equivalence relation.

Compositionality involves contexts, so the first step is to clarify the meaning of contexts. Given a set of intended observables (barbs) $Obs \subseteq \mathscr{N} \uplus \overline{\mathscr{N}}$, **a context should not provide barbs / observables**. If it does so, then every encoded term would for example exhibit barb $x \in Obs$, even $[\![\mathbf{0}]\!]$. This would violate and nullify the meaning of barbs, which is to distinguish different processes by observing the barbs. If every process exhibits the same barb, this barb then loses its purpose.

Also, **a context does not restrict barbs / observables**. Again, if a context restricts $x \in Obs$, we cannot distinguish $[\![x]\!]$ and $[\![\mathbf{0}]\!]$, which also contradicts with the purpose of barbs.

Finally, **a context does not depend on the process in its "hole"**. A context is fixed when the encoding is proposed, and requiring "there is an α in C if there is an α in p, and α becomes β if p has β instead of α", where C is a context and p is the process in its "hole", means we require a "variable" in the context, which does not exist in CCS and its extensions.

To prove the proposition, we first need to establish Lemma 2.

Lemma 2 The choice operator must be encoded into a choice operator in an encoding that is compositional, preserves divergence, and preserves semantics up to weak bisimulation or a finer equivalence relation.

Proof: Suppose the set of barbs include $\{y,z\}$. We first prove the lemma for weak bisimulation. For now $\asymp$ represents weak bisimulation.

Consider the process $y+z \in \mathscr{T}_{CCSS}$. $[\![y+z]\!] = C([\![y]\!],[\![z]\!])$ where C is a context (compositionality). Because $y+z$ is not divergent, $[\![y+z]\!]$ is also not divergent (Lemma 1). Then there is a finite computation:

$$[\![y+z]\!] = C([\![y]\!],[\![z]\!]) \longrightarrow\longrightarrow \cdots \longrightarrow T$$

where $T \stackrel{\tau}{\not\longrightarrow}$. For all such computations, $[\![y+z]\!] \Longrightarrow T$ implies that $y+z \Longrightarrow S$ where $S \asymp T$ by applying weak bisimilarity multiple times. Since $y+z$ does not reduce, we know that $y+z \Longrightarrow S$ represents no transitions, and as a result $S = y+z$. $S \xrightarrow{y}$ and $S \asymp T$ implies that $T \stackrel{\hat{y}}{\Longrightarrow}$ (weak bisimulation). Because T is the state we found on the computation that $T \stackrel{\tau}{\not\longrightarrow}$, we know that $T \stackrel{\hat{y}}{\Longrightarrow}$ consists no τ-transitions before y-transition and thus $T \xrightarrow{y}$, which means T barbs y directly. Analogously, $T \xrightarrow{z}$.

As stated above, the context does not provide barbs. $T \xrightarrow{y}$ and $T \xrightarrow{z}$ at the same time implies that $T = D'([\![y]\!],[\![z]\!])$ for some context D'. Otherwise, if T contains only one of $[\![y]\!]$ and $[\![z]\!]$, $T \xrightarrow{y} \wedge T \xrightarrow{z}$ indicates that the context in T provides y or z, and this contradicts with the requirements on contexts. This in turn suggests that the computation $C([\![y]\!],[\![z]\!]) \longrightarrow\longrightarrow \cdots \longrightarrow T$ does not depend on the content in the "holes", but merely on the context C, since $[\![y]\!]$ and $[\![z]\!]$ stay intact in T.

Figure 3 shows the parse tree of process T where each vertex represents an operator in CCS, the root is on top, and $[\![y]\!]$, $[\![z]\!]$ are each represented by a leaf.

The circle $\circ$ in Figure 3 represents the first operator that connects the sub-tree including $[\![y]\!]$ and $[\![z]\!]$, that is, we can write $T = D(C_1([\![y]\!]) \circ C_2([\![z]\!]))$, where D is the context with one "hole" that is represented by Figure 3 excluding the sub-tree with $\circ$ as root, C_1 and C_2 are contexts with one "hole" that is on the left / right side of $\circ$, respectively.

This means that we can write $T = D(C_1([\![y]\!]) \circ C_2([\![z]\!]))$

Considering $T \xrightarrow{y} \wedge T \xrightarrow{z}$, we know that no vertex in Figure 3 represents action prefixing, otherwise T cannot barb y and z directly, i.e. D, C_1, C_2 contain no action prefixing.

Now consider the process $y+\tau.z \in \mathscr{T}_{CCSS}$. $[\![y+\tau.z]\!] = C([\![y]\!],[\![\tau.z]\!])$ where C is the same context as above.

By applying weak bisimulation, the process graph of its encoding $[\![y+\tau.z]\!]$ is in Figure 4 (dashed arrows represent transitions with a non-silent label, double lined arrow is as in Definition 9).

Figure 3: Parse Tree of T

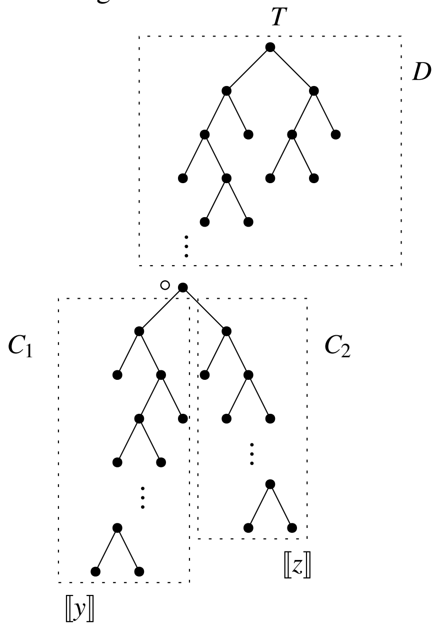
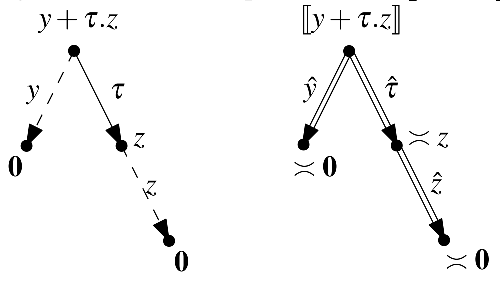


Figure 4: Process Graph of $y+\tau.z$, $[\![y+\tau.z]\!]$



As mentioned before, the computation from $C([\![y]\!],[\![z]\!])$ to $T = D(C_1([\![y]\!]) \circ C_2([\![z]\!]))$ only depends on C. Then $C([\![y]\!],[\![\tau.z]\!])$ can do a similar computation that only involves the context C to $R = D(C_1([\![y]\!]) \circ C_2([\![\tau.z]\!]))$, where D, C_1, C_2 contain no action prefixing.

Since $\tau.z \asymp [\![\tau.z]\!]$ (preservation of semantics) and $\tau.z \xrightarrow{\tau} z$, we have $[\![\tau.z]\!] \overset{\hat{\tau}}{\Longrightarrow} P$, $P \asymp z$, and subsequently $P \overset{\hat{z}}{\Longrightarrow} \asymp \mathbf{0}$ (weak bisimulation). Analogously, $[\![y]\!] \overset{\hat{y}}{\Longrightarrow} \asymp \mathbf{0}$.

Since the contexts C_1, C_2, D contain no action prefixing, and they cannot restrict y, z, or τ, we know that $C_1([\![y]\!])$ can do $C_1([\![y]\!]) \overset{\hat{y}}{\Longrightarrow}$, where C_1 possibly contributes some τ. ($P \overset{\hat{\alpha}}{\Longrightarrow}$ if and only if $\exists Q:\ P \overset{\hat{\alpha}}{\Longrightarrow} Q$.)

Similarly, $C_2([\![\tau.z]\!]) \overset{\hat{\tau}}{\Longrightarrow}\overset{\hat{z}}{\Longrightarrow}$.

Suppose the operator $\circ$ is a parallel composition operator, then there is a path:

$$C_1([\![y]\!]) \circ C_2([\![\tau.z]\!]) = C_1([\![y]\!]) | C_2([\![\tau.z]\!]) \overset{\hat{z}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}$$

because of (PAR-L) and (PAR-R). The interleaving semantics of the parallel composition in CCS means that the process $P|Q$ can always execute any α if P or Q can execute α.

Because D also does not restrict y, z, τ, there is a path $R = D(C_1(\llbracket y \rrbracket) | C_2(\llbracket \tau.z \rrbracket)) \overset{\hat{z}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}$. When we compose this path with the path from $\llbracket y+\tau.z \rrbracket$ to R, we have a new path:

$$\llbracket y+\tau.z \rrbracket = C(\llbracket y \rrbracket, \llbracket z \rrbracket) \Longrightarrow R \overset{\hat{z}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}$$

This means that $\llbracket y+\tau.z \rrbracket$ barbs both y and z, which contradicts with the process graph of $\llbracket y+\tau.z \rrbracket$ in Figure 4.

Since the other operators except for $+$ in CCS do not combine processes, the only possibility is that the $\circ$ is the choice operator $+$.

Now suppose $\asymp'$ is an equivalence relation finer than weak bisimulation, then for all P, $Q \in \mathscr{T}_{CCSS} \cup \mathscr{T}_{CCS}$, $P \asymp Q$ implies $P \asymp' Q$, which means every implication in the above proof involving weak bisimulation is also true for $\asymp'$, and the lemma is also true for $\asymp'$. $\square$

Lemma 3 The parallel composition operator must be encoded into a parallel composition operator in an encoding that is compositional and preserves semantics up to weak bisimulation or a finer equivalence relation.

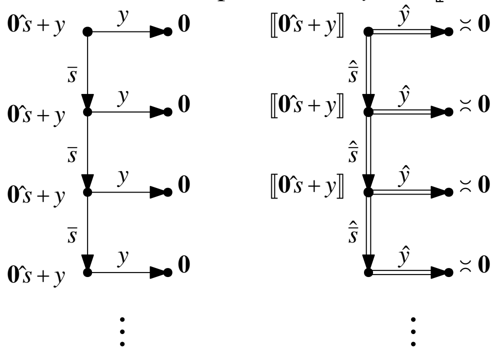
Proof: The proof can be obtained by analysing the process $y|z$ in a way similarly as in the proof for Lemma 2. $\square$

Proof: (of Proposition 5) Consider the process $\mathbf{0}\hat{\ }s+y$, with intended observables $\{y\}$, then $\llbracket \mathbf{0}\hat{\ }s+y \rrbracket = C(\llbracket \mathbf{0}\hat{\ }s \rrbracket, \llbracket y \rrbracket)) \Longrightarrow V$ where $V = D(C_1(\llbracket \mathbf{0}\hat{\ }s \rrbracket) + C_2(\llbracket y \rrbracket))$ with the same C, D, C_1, C_2 as in Lemma 2.

As mentioned earlier in the proof for Proposition 3, a process signalling s can be treated as it doing an action $\bar{s}$ and returning back to itself.

The process graphs are shown in Figure 5 (by applying weak bisimulation).

Figure 5: Process Graphs of $\mathbf{0}\hat{\ }s+y$ and $\llbracket \mathbf{0}\hat{\ }s+y \rrbracket$



Again by applying weak bisimulation, we can find a(n) (infinite) path $\llbracket \mathbf{0}\hat{\ }s \rrbracket \overset{\hat{\bar{s}}}{\Longrightarrow}\overset{\hat{\bar{s}}}{\Longrightarrow} \cdots$.

Suppose either of D, C_1 restricts s, $V = D(C_1(\llbracket \mathbf{0}\hat{\ }s \rrbracket) + C_2(\llbracket y \rrbracket))$, would not be able to execute s after a certain number of τ steps. At the same time, the context C does not provide s, otherwise we would require C to provide r if we consider the process $\llbracket \mathbf{0}\hat{\ }r+y \rrbracket$, and this means the context C depends on the process in its "hole", which contradicts with the meaning of contexts.

As a result, $\llbracket \mathbf{0}\hat{\ }r+y \rrbracket$ cannot weakly barb s, and this contradicts with the process graph of $\llbracket \mathbf{0}\hat{\ }s+y \rrbracket$ in Figure 5, and neither D nor C_1 restricts s.

Because C_1 and C_2 are connected with the choice operator, the process $C_1(\llbracket \mathbf{0}\hat{\ }s \rrbracket) + C_2(\llbracket y \rrbracket)$ has to choose between the lefthand and righthand side. Once a choice is made for $\bar{s}$, no more y is possible, and vice versa. Hence there is no such path like

$$C_1(\llbracket \mathbf{0}\hat{\ } s \rrbracket) + C_2(\llbracket y \rrbracket) \overset{\hat{\bar{s}}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow} \cdots$$

but only

$$C_1(\llbracket \mathbf{0}\hat{\ } s \rrbracket) + C_2(\llbracket y \rrbracket) \overset{\hat{\bar{s}}}{\Longrightarrow}\overset{\hat{\bar{s}}}{\Longrightarrow} \cdots, \text{ or } C_1(\llbracket \mathbf{0}\hat{\ } s \rrbracket) + C_2(\llbracket y \rrbracket) \overset{\hat{y}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow} \cdots$$

.

However, as shown in its process graph, $\llbracket \mathbf{0}\hat{\ } s + y \rrbracket$ has such path:

$$\llbracket \mathbf{0}\hat{\ } s + y \rrbracket = C(\llbracket \mathbf{0}\hat{\ } s \rrbracket, \llbracket y \rrbracket) \overset{\hat{\bar{s}}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}\overset{\hat{\bar{s}}}{\Longrightarrow} \cdots$$

which means that the context C (or D) provides $\bar{s}$ after $C_1(\llbracket \mathbf{0}\hat{\ } s \rrbracket) + C_2(\llbracket y \rrbracket)$ chooses y, and this contradicts with the requirements on contexts.

In conclusion, there is no path as

$$\llbracket \mathbf{0}\hat{\ } s + y \rrbracket = C'(D(C_1(\llbracket \mathbf{0}\hat{\ } s \rrbracket) + C_2(\llbracket y \rrbracket))) \overset{\hat{\bar{s}}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}\overset{\hat{\bar{s}}}{\Longrightarrow} \cdots$$

and this contradicts with the process graph of $\llbracket \mathbf{0}\hat{\ } s + y \rrbracket$.

□

Propositions 1 and 5 together establish the fact that CCSS is strictly more expressive than CCS; CCSS is separated from CCS by compositionality.

Proposition 6 There is no compositional (Criterion 1), divergence-reflecting (Criterion 4) encoding from CCSS to ABC that preserves semantics (Criterion 6) up to weak bisimulation (Definition 9) or a finer equivalence relation.

Proof: Again, the process $\llbracket y + \tau.z \rrbracket$ can execute a series of reductions and reach a state $T' = D(C_1(\llbracket y \rrbracket) \circ C_2(\llbracket \tau.z \rrbracket))$ with the same contexts as before.

Suppose the $\circ$ is parallel compositon. Because there is no action prefixing in either C_1 or C_2, the effect of broadcast mechanism has no effect on the proof. For example, even if there is such processes like $(b! | \llbracket y \rrbracket) | \llbracket \tau.z \rrbracket + b?$ or $(b! + \llbracket y \rrbracket) | \llbracket \tau.z \rrbracket + b?$, we can still find a path that includes $\overset{\hat{z}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}$, since the broadcast actions are not prefix of either $\llbracket y \rrbracket$ or $\llbracket \tau.z \rrbracket$. As a result, the $\circ$ must be a choice operator and the proof for Proposition 5 is still valid for ABC. □

Proposition 7 There is no compositional (Criterion 1), divergence-reflecting (Criterion 4) encoding from CCSS to CCS^{sg} that preserves semantics (Criterion 6) up to weak bisimulation (Definition 9) or a finer equivalence relation.

Proof: In CCS^{sg}, only silent actions with higher priority can preempt the ecexution of other actions. Again, the process $\llbracket y + \tau.z \rrbracket$ reduces to a state $T = D(C_1(\llbracket y \rrbracket) \circ C_2(\llbracket \tau.z \rrbracket))$ and there is no outgoing transitions labelled with any silent action. As a result, the priority mechanism plays no rule here. Suppose $\circ$ is a paralle composition, without any preemption, $C_1(\llbracket y \rrbracket)$ and $C_2/\llbracket \tau.z \rrbracket$ exhibiits interleaving semantics and we can find a path that includes $\overset{\hat{z}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}$. Again, by composing the two paths, we acquire a new path: $\llbracket y + \tau.z \rrbracket \overset{T}{\Longrightarrow}\overset{\hat{z}}{\Longrightarrow}\overset{\hat{y}}{\Longrightarrow}$ and this leads to contradiction, and the choice operator must be encoded into a choice operator.

The proof hereafter is identical with the proof for Proposition 5. Note that the effect of priorities makes no distinction, because it only adds more conditions to the SOS-rules of the choice operator, but once a choice is made for the left hand side of $C_1(\llbracket \mathbf{0}\hat{\ } s \rrbracket) + C_2(\llbracket y \rrbracket)$, it still becomes impossible to continue barbing y. □

In conclusion, CCSS cannot be compositionally encoded into CCS, ABC, CCS^{sg} up to a semantic-preserving encoding w.r.t. weak bisimulation or a finer relation.

3.4 ABC to CCS and CCSS

In [1], Pohjola and Parrow separated *monotonic composition* and *non-monotonic composition*. In this thesis, we use *monotonic parallel composition* to separate ABC from CCS and CCSS.

Definition 14 (Monotonic Parallel Composition) A parallel composition $| : \mathscr{T} \times \mathscr{T} \to \mathscr{T}$ is monotonic, if $\forall P, Q, R \in \mathscr{T} :\ P \Longrightarrow Q$ implies $P|R \Longrightarrow Q|R$.

A parallel composition is *non-monotonic*, if it is not monotonic.

The parallel composition of CCS(S) is monotonic, which is easily derived from rules (PAR-L) and (PAR-R) in Table 2. In ABC it is not, due to the restriction in Table 5, rules (BRO-L) and (BRO-R).

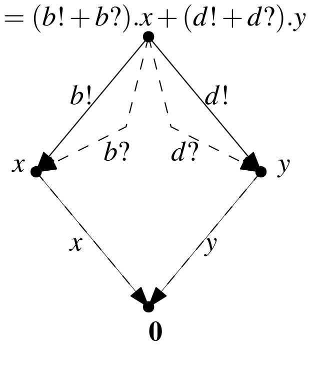
Proposition 8 a) There is no homomorphic (Criterion 2), divergence-reflecting (Criterion 4), and weakly observation-respecting (Criterion 5.2) encoding from ABC to CCS;

b) There is no homomorphic, barb preserving (Criterion 7), and operational correspondent (Criterion 3) encoding from ABC to CCS, up to an equivalence that preserves barb (Definition 12).

Proof: By contradiction.
Part a): homomorphic, divergence-reflecting, and weakly observation-respecting.

Reasonably, the intended observables are $Obs := \mathscr{A}_{ABC} \backslash (\{\tau\} \cup \mathscr{B}! \cup \mathscr{B}?)$. Consider the process $P = (b! + b?).x + (d! + d?).y$, the process graph of P is shown in Figure 6 (note that the reduction relation is defined over $\{\tau\} \cup \mathscr{B}!$, dashed arrows represent non-reduction transitions):

Figure 6: Process Graph of P



The process can make the choice between actions over channel b or channel d, exposing x or y, respectively. Once a choice is made between b and d, either x or y is exposed, not at the same time.

There are only two maximal computations of P:

$$\pi_1 : P \longrightarrow x;\ \pi_2 : P \longrightarrow y$$

with $Obs(\pi_1) = \{x\}$, $Obs(\pi_2) = \{y\}$. Thus by Criterion 5.2 there exists maximal computations π_1', π_2' of $[\![P]\!]$, such that $Obs(\pi_1') = \{x\} \wedge Obs(\pi_2') = \{y\}$. Suppose they are:

$$\pi_1' : [\![P]\!] \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots \longrightarrow s_n;\ \pi_2' : [\![P]\!] \longrightarrow t_1 \longrightarrow t_2 \longrightarrow \cdots \longrightarrow t_m$$

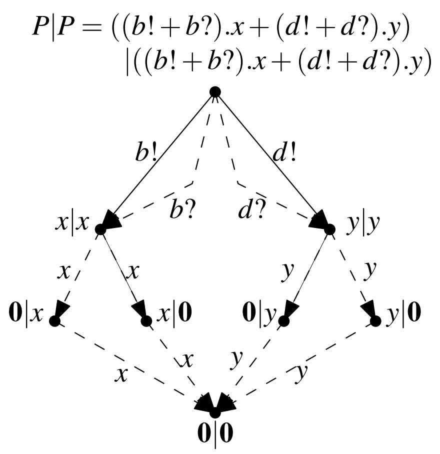
We know that π_1', π_2' are finite because $[\![P]\!]$ is not divergent (P is not divergent, Lemma 1).

$Obs(\pi_1') = \{x\}$ means that $\exists s \in \{[\![P]\!], s_1, s_2, \dots s_n\} : s \xrightarrow{x}$ (Definition 4).

Similarly, $\exists t \in \{[\![P]\!], t_1, t_2, \dots t_m\} : t \xrightarrow{y}$.

Now we look at the process $P|P$. Its process graph is shown in Figure 7:

Figure 7: Process Graph of $P|P$



There are also only two maximal computations of $P|P$:

$$\pi_3 : P|P \longrightarrow x|x;\ \pi_4 : P|P \longrightarrow y|y$$

with $Obs(\pi_3) = \{x\} \wedge Obs(\pi_4) = \{y\}$.

The encoding of $P|P$ however exhibits more barbs. $[\![P|P]\!] = [\![P]\!]|[\![P]\!]$ (Criterion 2), and we can compose computations π_1', π_2' together and acquire a new computation:

$$\pi_5 : [\![P|P]\!] = [\![P]\!]|[\![P]\!] \longrightarrow s_1|[\![P]\!] \longrightarrow s_2|[\![P]\!] \longrightarrow \cdots \longrightarrow s_n|[\![P]\!] \longrightarrow s_n|t_1 \longrightarrow s_n|t_2 \longrightarrow \cdots \longrightarrow s_n|t_m$$

For $P, Q \in \mathscr{T}_{CCS}$, if $P \xrightarrow{x}$, then $P|Q \xrightarrow{x}$ and $Q|P \xrightarrow{x}$ ((PAR-L), (PAR-R)).

Thus $\exists s \in \{[\![P]\!]|[\![P]\!], s_1|[\![P]\!], s_2|[\![P]\!], \dots s_n|[\![P]\!]\} : s \xrightarrow{x}$ and $\exists t \in \{s_n|[\![P]\!], s_n|t_1, s_n|t_2, \dots s_n|t_m\} : t \xrightarrow{y}$, which means $Obs(\pi_5) = \{x, y\}$, not a subset of either $Obs(\pi_3)$ or $Obs(\pi_4)$. $\lightning$ (Criterion 5.2)

Part b): homomorphic, barb preserving, operational correspondent up to an barb preserving equivalence.

Again consider the processes P and $P|P$ with the set of observables that include $\{x, y\}$.

$[\![P|P]\!] = [\![P]\!]|[\![P]\!]$ (homomorphy).

$P \Longrightarrow x$ implies $[\![P]\!] \Longrightarrow S \wedge S \asymp [\![x]\!]$ for some S (completeness). Suppose the path is

$$\pi_1 : [\![P]\!] \longrightarrow \cdots \longrightarrow S$$

$P \Longrightarrow y$ implies $[\![P]\!] \Longrightarrow T$ where $T \asymp [\![y]\!]$, with path

$$\pi_2 : [\![P]\!] \longrightarrow \cdots \longrightarrow T$$

We can compose π_1 and π_2 into

$$\pi_3 : [\![P|P]\!] = [\![P]\!]|[\![P]\!] \longrightarrow \cdots \longrightarrow S|[\![P]\!] \longrightarrow \cdots \longrightarrow S|T$$

$x \downarrow x$ implies $[\![x]\!] \downarrow_{\varphi_{[\![]\!]}(x)}$ (barb preservation of $([\![\ ]\!], \varphi_{[\![]\!]})$), thus $S \downarrow_{\varphi_{[\![]\!]}(x)}$ (barb preservation of $\asymp$). Similarly, $T \downarrow_{\varphi_{[\![]\!]}(y)}$.

$[\![x]\!] \downarrow_{\varphi_{[\![]\!]}(x)}$ and $[\![y]\!] \downarrow_{\varphi_{[\![]\!]}(y)}$ implies $[\![x]\!]|[\![y]\!] \downarrow_{\varphi_{[\![]\!]}(x)}$ and $S|T \downarrow_{\varphi_{[\![]\!]}(y)}$: for all CCS (CCSS) processes P and Q, if $P \downarrow x$ and $Q \downarrow y$, then $P \xrightarrow{x}$ and $Q \xrightarrow{y}$ (Definition 4), then $P|Q \xrightarrow{x}$, $P|Q \xrightarrow{y}$, $Q|P \xrightarrow{x}$, and $Q|P \xrightarrow{y}$ ((PAR-L), (PAR-R)), which means $P|Q \downarrow x$ and $P|Q \downarrow y$:

In conclusion, $[\![P|P]\!] \Longrightarrow\asymp [\![x|y]\!]$, and this implies $P|P \Longrightarrow Q \wedge [\![Q]\!] \asymp [\![x|y]\!]$ for some Q (SOUND), and $[\![Q]\!] \downarrow_{\varphi_{[\![]\!]}(x)} \wedge [\![Q]\!] \downarrow_{\varphi_{[\![]\!]}(y)}$ (barb preservation of $\asymp$). This in turn implies $Q \downarrow x \wedge Q \downarrow y$ (barb preservation of $\asymp$).

This means that $P|P$ reduces to a state that barbs x and y at the same time, which contradicts the process graph of $P|P$. $\lightning$

□

These separation results rely on the fact that broadcast actions enforce all the willing parts to participate, while the parallel composition operator in CCS allows the participants to act independently, a.k.a. the parallel operator is monotonic. This result extends to CCSS, as CCSS also has monotonic parallel composition.

Proposition 9 The separation results extends to CCSS.

Proof: Same as Proposition 8 and ommited. □

Proposition 8 and Proposition 9 shows that ABC is separated from CCS and CCSS by homomorphy.

3.5 ABC to CCS^{sg}

Proposition 10 There is a homomorphic (Criterion 2) encoding (w.r.t. all operators except prefixing for broadcast and receive actions) from ABC to CCS^{sg} (with 3 priorities) that respects operational correspondence (Criterion 3) up to identity function, and divergence reflection (Criterion 4).

Proof: For readability, we write α_h for $\alpha{:}0$, α_m for $\alpha{:}1$, α_l for $\alpha{:}2$, meaning high, medium, low priority, respectively.

The encoding $([\![\ ]\!], \varphi_{[\![]\!]})$ is as follows[4] ($b \in \mathscr{B}$, $\eta \in \mathscr{H}_{ABC} \uplus \{\tau\}$):

$$\begin{array}{lll}
[\![b?.P]\!] := b_h.\tau_m.[\![P]\!] & [\![b!.P]\!] := \tau_l.(\overline{b_h}.X + \tau_m.[\![P]\!]) \text{ where } X \overset{def}{=} \overline{b_h}.X + \tau_m.[\![P]\!] & \\
[\![\eta.P]\!] := \eta.[\![P]\!] & [\![P|Q]\!] := [\![P]\!]|[\![Q]\!] & [\![\sum_{i\in I} P_i]\!] := \sum_{i\in I}[\![P_i]\!] \ (I \text{ an index set}) \\
[\![P[f]]\!] := [\![P]\!][f'] & [\![P\backslash L]\!] := [\![P]\!]\backslash\varphi_{[\![]\!]}(L) & [\![A]\!] := P' \text{ for } A \overset{def}{=} P, \text{ and } P' \overset{def}{=} [\![P]\!]
\end{array}$$

The renaming policy $\varphi_{[\![]\!]}(x) := \begin{cases} x_h, \ \forall x \in \mathscr{B} \\ x_l, \ \forall x \in \mathscr{N}_{CCS} \end{cases}$ is injective (please refer to Definition 7), and it extends to $\mathscr{A}_{ABC}$ by defining $\varphi_{[\![]\!]}(\overline{x}) := \overline{\varphi_{[\![]\!]}(x)}$ ($x \in \mathscr{N}_{CCS}$), $\varphi_{[\![]\!]}(x?) := x_h$, $\varphi_{[\![]\!]}(x!) := \overline{x_h}$ ($x \in \mathscr{B}$). The extended renaming policy $\varphi_{[\![]\!]} : \mathscr{A}_{ABC} \to \mathscr{A}_{ABC}$ is still injective. Although both $x?$ and $x!$ are renamed to the same channel x_h, they are renamed into exclusively input or output actions, and this preserves the injectivity of the renaming policy.

[4]The encoding was originally proposed by Johannes Åman Pohjola (Johannes.Amanpohjola@data61.csiro.au) during an afternoon tea session, and the proofs are done by the author.

The agent identifiers X and P' are *fresh*, in the sense that every encoding of a process including broadcast action prefix or agent identifier generates a different X or P', respectively. For all $\alpha \in \mathscr{A}_{ABC}$, the relabelling function f' is defined as $\begin{cases} f'(\alpha) := \alpha, & \text{if } \alpha \notin image(\varphi_{[\![]\!]}); \\ f'(\varphi_{[\![]\!]}(\alpha)) := \varphi_{[\![]\!]}(f(\alpha)), & \text{else.} \end{cases}$

$\varphi_{[\![]\!]}(L) := \{\varphi_{[\![]\!]}(\alpha) \mid \alpha \in L\}$.

$[\![\mathbf{0}]\!] = [\![\Sigma_{i\in\emptyset} P_i]\!] = \Sigma_{i\in\emptyset}[\![P_i]\!] = \mathbf{0}$.

The encoding is homomorphic for all operators except prefixing with $b!$ and $b?$ per definition. Its operational correspondence is established with Lemmas 4, and 12, and its divergence reflection is proved with Lemma 13.

□

Lemma 4 For all $S \in \mathscr{T}_{ABC}$, if $S \Longrightarrow S'$ then $[\![S]\!] \Longrightarrow [\![S']\!]$. (The $\asymp$ symbol is left out because it is the identity function in this case, i.e. $P \asymp Q$ iff $P = Q$.)

Proof: This lemma is trivially true if $S = S'$, because the relation $\Longrightarrow$ is reflexive. If $S \neq S'$, we can write the path (computation) explicitly:

$$S \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots s_{n-1} \longrightarrow S'$$

for some $s_i \in \mathscr{T}_{ABC}$, $1 \leq i \leq n-1$, $i \in \mathbb{N}$.

Applying Lemma 5 n times, there is a corresponding path in CCS^{sg} with 3 priorities:

$$[\![S]\!] \Longrightarrow [\![s_1]\!] \Longrightarrow [\![s_2]\!] \Longrightarrow \cdots [\![s_{n-1}]\!] \Longrightarrow [\![S']\!]$$

thus $[\![S]\!] \Longrightarrow [\![S']\!]$. □

Lemma 5 For all $S \longrightarrow S'$, it holds that $[\![S]\!] \Longrightarrow [\![S']\!]$.

Proof: If we explicitly label the transition as $S \xrightarrow{\alpha} S'$, then there are two possibilities (note in ABC the set of reduction labels is $\mathscr{B}! \uplus \{\tau\}$):

Case 1: $\alpha = \tau$, this lemma is directly implied from Lemma 8, whose proof is built upon Lemma 6, and 7.

Case 2: $\alpha \in \mathscr{B}!$, let $j = 0$ in Lemma 11 then we have the proof. For Lemma 11 we need Lemma 10.

□

Lemma 6 If P is an ABC process, then $\nexists P',\ P'' \in \mathscr{T}_{CCS^{sg}} :\ [\![P]\!] \xrightarrow{\tau_h} P' \vee [\![P]\!] \xrightarrow{\tau_m} P''$, which means $[\![P]\!]$ cannot preempt the execution of any action with priority l.

Proof: By contradiction. Suppose $[\![P]\!]$ can preempt the execution of some action, which means $[\![P]\!] \xrightarrow{\tau_m} P'$ or $[\![P]\!] \xrightarrow{\tau_h} P''$ for some $P', P'' \in \mathscr{T}_{CCS^{sg}}$ (Definition 6).

From the encoding we can see: τ_m always has at least a τ_l, b_h, or $\overline{b_h}$ as prefix, thus there is no outgoing transition labelled with τ_m from any $[\![P]\!]$.

Also, as shown in the encoding, there is no τ_h as subterm in $[\![P]\!]$ for any P; there are only b_h and $\overline{b_h}$ for some $b \in \mathscr{B}$ (please refer to Section 2.4 for names in CCS^{sg}), which come from the encoding of $b?$ or $b!$, respectively.

As a result, the transition $[\![P]\!] \xrightarrow{\tau_h} S$ for some S must involve the handshake communication of a pair of high-prioritised input and output actions. However, any $\overline{b_h}$ for $b \in \mathscr{B}$ has at least a τ_l as prefix, which means any transition involving $\overline{b_h}$ must follow (at least) one transition labelled with τ_l.

Thus any handshake communication resulting in τ_h must be a successor of (at least) one transition labelled with τ_l, and there is no such transition: $[\![P]\!] \xrightarrow{\tau_h} S$ for any $P \in \mathscr{T}_{ABC}$ and $S \in \mathscr{T}_{CCS^{sg}}$.

□

Lemma 7 If S is an ABC process, for all $S \xrightarrow{c} S'$ where $c \in \mathscr{H}_{CCS}$, it holds that $[\![S]\!] \xrightarrow{c_l} [\![S']\!]$.

Proof: Trivially, the lemma holds for $S = \mathbf{0}$. Assume the lemma holds for some P, Q, P_i ($i \in I$, I an index set). We prove by structural induction that it also holds for processes that are constructed using P, Q, P_i ($i \in I$) and operators in ABC.

- *Prefixing:* Suppose $S = c.P$, then $[\![S]\!] = c_l.[\![P]\!]$. $S \xrightarrow{c} S' \wedge S = c.P$ implies $S' = P$, and since $[\![S]\!] = c_l.[\![P]\!]$, we have $[\![S]\!] \xrightarrow{c_l} [\![P]\!] = [\![S']\!]$.
- *Choice:* Suppose $S = \sum_{i\in I} P_i$, then $[\![S]\!] = \sum_{i\in I}[\![P_i]\!]$. Without loss of generality, we can assume $S \xrightarrow{c} S'$ derived from $P_j \xrightarrow{c} P'$ where $S' = P'$, $j \in I$. Since the lemma holds for P_j, we know that $[\![P_j]\!] \xrightarrow{c_l} [\![P']\!]$, and thus $[\![S]\!] \xrightarrow{c_l} [\![P']\!] = [\![S']\!]$.
- *Parallel composition:* Suppose $S = P|Q$, $S \xrightarrow{c} S'$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$. Because $c \neq \tau$, the transition stems inevitably from either P or Q. Suppose the transition stems from $P \xrightarrow{c} P'$, thus $S' = P'|Q$ and $[\![P]\!] \xrightarrow{c_l} [\![P']\!]$. Now we can compose a path:

  $$[\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{c_l} [\![P']\!]|[\![Q]\!] = [\![S']\!]$$

  considering that $[\![Q]\!]$ cannot preempt c_l (Lemma 6).
- *Agent Identifier:* Suppose $S \overset{def}{=} P$, then $S \xrightarrow{c} S'$ if and only if $P \xrightarrow{c} P'$ with $P' = S'$ ((REC) in Table 2), which implies $[\![P]\!] \xrightarrow{c_l} [\![P']\!]$. Since $[\![S]\!] = P''$ where $P'' \overset{def}{=} [\![P]\!]$, we can infer $P'' \xrightarrow{c_l} [\![P']\!]$ and thus $[\![S]\!] \xrightarrow{c_l} [\![P']\!] = [\![S']\!]$ (SOS-rule for agent identifier in Table 7).
- *Relabelling:* Suppose $S = P[f]$. $S \xrightarrow{c} S'$ implies $\exists a \in \mathscr{N}_{ABC}: (f(a) = c) \wedge (P \xrightarrow{a} P') \wedge (S' = P'[f])$, otherwise the transition $S \xrightarrow{c} S'$ cannot be derived from any rule in Table 2, which is a part of ABC semantics. The relabelling function in ABC is defined as $f : (\mathscr{B} \rightarrow \mathscr{B}) \cup (\mathscr{N}_{CCS} \rightarrow \mathscr{N}_{CCS})$ (Section 2.3), it maps handshake names to handshake names, thus $a \in \mathscr{N}_{CCS}$ only when $c \in \mathscr{N}_{CCS}$. Due to the induction hypothesis, $P \xrightarrow{c} P'$ implies $[\![P]\!] \xrightarrow{a_l} [\![P']\!]$.

  $[\![S]\!] = [\![P[f]]\!] = [\![P]\!][f']$ and $\varphi_{[\![]\!]}(f(a)) = f'(\varphi_{[\![]\!]}(a))$.

  $[\![P]\!] \xrightarrow{a_l} [\![P']\!]$ implies $[\![P]\!][f'] \xrightarrow{f'(a_l)} [\![P']\!][f']$ (Table 2 (REL)), where a_l is nothing special other than $\varphi_{[\![]\!]}(a)$, and we might as well write it this way:

  $$f'(a_l) = f'(\varphi_{[\![]\!]}(a)) \overset{\text{per definition}}{=} \varphi_{[\![]\!]}(f(a)) \overset{\text{f(a)=c}}{=} \varphi_{[\![]\!]}(c) = c_l.$$

  As a result, $[\![S]\!] = [\![P[f]]\!] = [\![P]\!][f'] \xrightarrow{c_l} [\![P']\!][f'] = [\![P'[f]]\!] = [\![S']\!]$.
- *Restriction:* Suppose $S = P\backslash L$, $S \xrightarrow{c} S'$ implies $P \xrightarrow{c} P'$, $c, \bar{c} \notin L$ and $S' = P'\backslash L$, otherwise the transition $S \xrightarrow{c} S'$ cannot be derived from any rule in Table 2. Thus $[\![P]\!] \xrightarrow{c_l} [\![P']\!]$, where $c_l = \varphi_{[\![]\!]}(c)$. $\varphi_{[\![]\!]}(L)$ is defined as $\varphi_{[\![]\!]}(L) := \{\varphi_{[\![]\!]}(\alpha) \mid \alpha \in L\}$. Because $\varphi_{[\![]\!]}()$ is injective, $c, \bar{c} \notin L$ implies $\varphi_{[\![]\!]}(c), \varphi_{[\![]\!]}(\bar{c}) \notin \varphi_{[\![]\!]}(L)$. Thus $[\![P]\!] \xrightarrow{\varphi_{[\![]\!]}(c)} [\![P']\!]$ implies $[\![P]\!]\backslash\varphi_{[\![]\!]}(L) \xrightarrow{\varphi_{[\![]\!]}(c)} [\![P']\!]\backslash\varphi_{[\![]\!]}(L)$, which is identical with $[\![P\backslash L]\!] \xrightarrow{c_l} [\![P'\backslash L]\!]$ and $[\![S]\!] \xrightarrow{c_l} [\![S']\!]$.

□

Lemma 8 If S is an ABC process, for all $S \xrightarrow{\tau} S'$, it holds that $[\![S]\!] \xrightarrow{\tau_l} [\![S']\!]$.

Proof: By structural induction.

Replacing c by τ in the proof for Lemma 7, we have the proof for all cases except for transitions originated from handshake communication (COMM): $S = P|Q \xrightarrow{\tau} S'$. We can assume without loss of generality that for some $c \in \mathscr{H}_{CCS}$:

$$S \xrightarrow{\tau} S' \text{ because } P \xrightarrow{c} P',\ Q \xrightarrow{\overline{c}} Q' \wedge S' = P'|Q'$$

From Lemma 7 we know that $[\![P]\!] \xrightarrow{c_l} [\![P']\!]$, $[\![Q]\!] \xrightarrow{\overline{c_l}} [\![Q']\!]$. Because either the encoding of P or Q can preempt the execution of any action with priority l (Lemma 6), $[\![P]\!]$ and $[\![Q]\!]$ can communicate on low-priority channel and form a silent action:

$$[\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{\tau_l} [\![P']\!]|[\![Q']\!] = [\![S']\!]$$

□

Lemma 9 For all $P \in \mathscr{T}_{CCS^{sg}}$, if $P \xrightarrow{\alpha_m}$ or $P \xrightarrow{\alpha_l}$ then $P \overset{\tau_h}{\not\longrightarrow}$ for all $\alpha \in \mathscr{N}_{CCS}$.

Proof: First we need to prove that $P \xrightarrow{\tau_h}$ implies $\tau_h \in I(P)$.

$P \xrightarrow{\tau_h}$ means that

i) τ_h appears as subterm in P; (exclusive) or

ii) there is at least a pair of c_h and $\overline{c_h}$ in P for some $c \in \mathscr{N}_{CCS}$; (exclusive) or

iii) both.

Otherwise we cannot find an SOS-rule in Table 7 that results in a transition labelled with $\tau_h = \tau{:}0$.

As we can see from Table 6, if $\alpha{:}k$ is a subterm of P, then it is always an element of $I(P)$ except when

- all occurrances of $\alpha{:}k$ are restricted in P, i.e. $\alpha{:}k$ is removed from $I(P)$ by set subtraction during the construction of $I(P)$; or
- all occurrances of $\alpha{:}k$ are renamed to some name other than itself.

However, when $\alpha{:}k = \tau_h = \tau{:}0$, it can never be restricted, and it is always renamed to itself (please refer to 2.4). Therefore τ_h is always an element of $I(P)$ if it appears as a subterm in P, and this proves both i) and iii).

In case of ii), since there is no τ_h as subterm in P, $P \xrightarrow{\tau_h}$ must stem from handshake communication, which means we must be able to find $P_1|P_2$ as subterm of P that satisfies $P_1 \xrightarrow{c_h}$ and $P_2 \xrightarrow{\overline{c_h}}$ for some c. This implies that c is not restricted in P_1 and P_2, and / or $c = f(a)$ for some a.

Either way, $c \in I(P_1)$ and $\overline{c} \in I(P_2)$, which in turn implies that $\tau_h \in I(P_1|P_2)$, where $P_1|P_2$ is a subterm of P.

With the same reasoning as before, τ_h cannot be restricted and is always renamed to itself, thus τ_h is an element of $I(P)$.

Then we proceed to prove that "$\forall \alpha_l, \alpha_m \in \mathscr{A}_{CCS^{sg}}, P \in \mathscr{T}_{CCS^{sg}}$: if $P \xrightarrow{\alpha_l}$ or $P \xrightarrow{\alpha_m}$ then $P \overset{\tau_h}{\not\longrightarrow}$", which is equivalent with "$\forall \alpha_l, \alpha_m \in \mathscr{A}_{CCS^{sg}}, P \in \mathscr{T}_{CCS^{sg}}$: if $P \xrightarrow{\tau_h}$ then $P \overset{\alpha_l}{\not\longrightarrow}$ and $P \overset{\alpha_m}{\not\longrightarrow}$".

We can again prove it by structural induction. For instance, in case of parallel composition, if $P = P_1|P_2 \xrightarrow{\tau_h}$ stems from left interleaving, then $\tau_h = \tau{:}0 \in I(P_1)$. $I(P_1|P_2) = I(P_1) \cup I(P_2) \cup \{\tau{:}0 \mid I(P_1) \cap \overline{I(P_2)} \neq \emptyset\}$ (Table 6), $I(P_1|P_2)$ is a superset of $I(P_1)$, which implies that $\tau_h \in I(P_1|P_2) = I(P)$, hence P does not do any action with priority lower than 0. The reasoning for other cases is similar and is omitted.

□

Lemma 10 If S is an ABC process, for all $S \xrightarrow{b?} S'$ where $b? \in \mathscr{B}?$, there exists $n \in \mathbb{N}^+$ such that $[\![S]\!] \xrightarrow{b_h}{}^n \xrightarrow{\tau_m}{}^n [\![S']\!]$, where $\xrightarrow{\alpha}{}^n$ is n transitions sequentially composed: $\xrightarrow{\alpha}\xrightarrow{\alpha}\cdots\xrightarrow{\alpha}$.

Proof: By structural induction. The lemma is again trivially true for $\mathbf{0}$. Suppose it is also true for some P,Q,P_i (I an index set).

- *Prefixing:* Suppose $S = b?.P$, there is only one outgoing transition from S labelled with broadcast action: $S \xrightarrow{b?} P$, which means $S' = P$. $[\![S]\!] = b_h.\tau_m.[\![P]\!]$, there is a path: $[\![S]\!] \xrightarrow{b?} \tau_m.[\![P]\!] \xrightarrow{\tau_m} [\![P]\!] = [\![S']\!]$. Let $n = 1$ then the lemma is true. If $S = \alpha.P$ where $\alpha \notin \mathscr{B}?$ then there is no outgoing transition from S with a receive label, the condition in the lemma does not hold, hence the lemma is true.
- *Choice:* Similarly as in the proof for Lemma 7.
- *Parallel composition:* Suppose $S = P|Q$, $S \xrightarrow{b?} S'$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$.
  - If the transition stems from interleaving ((PAR-L), (PAR-R), (BRO-L), or (BRO-R)), we can again compose a path where $S' = P'|Q$ or $S' = P|Q'$ and prove the lemma, just as we did in the proof for Lemma 7.
  - If the transition stems from a broadcast-synchronisation ((BRO-C)), which means $S \xrightarrow{b?} S'$ stems from both $P \xrightarrow{b?} P'$ and $Q \xrightarrow{b?} Q'$, with $S' = P'|Q'$. Because the lemma is true for P and Q by induction hypothesis, there are two paths:

    $$\pi_1 : [\![P]\!] \xrightarrow{b_h}{}^i R \xrightarrow{\tau_m}{}^i [\![P']\!];\ \pi_2 : [\![Q]\!] \xrightarrow{b_h}{}^j T \xrightarrow{\tau_m}{}^j [\![Q']\!].$$

    where R and T are intermediate states on π_1 and π_2. $R \xrightarrow{\tau_m}$ means $R \not\xrightarrow{\tau_h}$ (Lemma 9), and the same for T. This means that neither R nor S can preempt an action with medium priority. Now we can compose a third path:

    $$\pi_3 : [\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{b_h}{}^i R|[\![Q]\!] \xrightarrow{b_h}{}^j R|T \xrightarrow{\tau_m}{}^i \xrightarrow{\tau_m}{}^i [\![P']\!]|[\![Q']\!] = [\![S']\!]$$

    Let $n = i + j$, the lemma is then also true for S.
  - *Agent Identifier:* Suppose $S \overset{def}{=} P$, the proof is similar to the proof for Lemma 7.
  - *Relabelling:* Suppose $S = P[f]$, $S \xrightarrow{b?} S'$, then there exists d such that $f(d) = b$ and $P \xrightarrow{d?} P'$ with $S' = P'[f]$, as stated in Lemma 7. Thus $[\![P]\!] \xrightarrow{d_h}{}^n \xrightarrow{\tau_m}{}^n [\![P']\!]$ for some n. Explicitly writing the path:

    $$[\![P]\!] \xrightarrow{d_h} p_1 \xrightarrow{d_h} \cdots \xrightarrow{d_h} p_n \xrightarrow{\tau_m} p_{n+1} \xrightarrow{\tau_m} \cdots \xrightarrow{\tau_m} p_{2n-1} \xrightarrow{\tau_m} [\![P']\!]$$

    Again, since $f'(d_h) = f'(\varphi_{[\![]\!]}(d)) = \varphi_{[\![]\!]}(f(d)) = \varphi_{[\![]\!]}(b) = b_h$, and $f'(\tau_m) = \tau_m$ (considering τ_m is not in the image of $\varphi_{[\![]\!]}()$), we can apply Rule (REL) $2n$ times and have the resulting path:

    $$[\![P]\!][f'] \xrightarrow{b_h} p_1[f'] \xrightarrow{b_h} \cdots \xrightarrow{b_h} p_n[f'] \xrightarrow{\tau_m} p_{n+1}[f'] \xrightarrow{\tau_m} \cdots \xrightarrow{\tau_m} p_{2n-1}[f'] \xrightarrow{\tau_m} [\![P']\!][f']$$

    Because $[\![S]\!] = [\![P[f]]\!] = [\![P]\!][f']$ and $[\![S']\!] = [\![P'[f]]\!] = [\![P']\!][f']$, we have the proof.
  - *Restriction:* Suppose $S = P\backslash L$, $S \xrightarrow{b?} S'$ implies $P \xrightarrow{b?} P'$ with $S' = P'\backslash L$ (broadcast names cannot be restricted).
    Again, $P \xrightarrow{b?} P'$ implies $[\![P]\!] \xrightarrow{b_h}{}^n \xrightarrow{\tau_m}{}^n [\![P']\!]$. Since $b_h,\ \tau_m \notin \varphi_{[\![]\!]}(L)$. Applying (RES) $2n$ times, we have a new path

    $$[\![S]\!] = [\![P\backslash L]\!][\![P]\!]\backslash\varphi_{[\![]\!]}(L) \xrightarrow{b_h}{}^n \xrightarrow{\tau_m}{}^n [\![P']\!]\backslash\varphi_{[\![]\!]}(L) = [\![P'\backslash L]\!] = [\![S']\!]$$

    and we have the proof.

□

Lemma 11 For all $j \in \mathbb{N}$, if $P \xrightarrow{b!} P'$, then there exists $n \in \mathbb{N}$ such that $[\![P]\!] \xrightarrow{\tau_l}\xrightarrow{\tau_h}{}^{n}\xrightarrow{\overline{b_h}}{}^{j}\xrightarrow{\tau_m}{}^{n+1} [\![P']\!]$.

Informally, this lemma states that the encoding translates one broadcast transition into a series of arbitrarily many $\overline{b_h}$ actions following and followed by a series of silent actions. n corresponds to the number of $b?$ that participated in the transition $P \xrightarrow{b!} P'$.

Proof: The lemma is again trivially true for $\mathbf{0}$. Suppose it is true for some processes P, Q, P_i ($i \in I$, I an index set), we prove it by structural induction.

- *Prefixing:* Suppose $S = b!.P$, thus $[\![S]\!] = \tau_l.(\overline{b_h}.X + \tau_m.[\![P]\!])$ where $X \overset{def}{=} \overline{b_h}.X + \tau_m.[\![P]\!]$. After executing τ_l, the X provides an arbitrary number of b_h actions, before executing a τ_m and reach the state $[\![P]\!]$. Let $n = 0$, we have $S' = P'$ and the lemma is true for such S.
- *Choice:* Same as in previous proofs.
- *Parallel composition:* Suppose $S = P|Q$, $S \xrightarrow{b!} S'$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$.
  - Suppose the transition stems from interleaving. The proof is again the same as in Lemma 7.
  - 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 $Q \xrightarrow{b!} Q'$. Without loss of generality, we can assume it to be the first case.

    Lemma 10 implies that there exists $k > 0$ such that $[\![Q]\!] \xrightarrow{b_h}{}^{k} T \xrightarrow{\tau_m}{}^{k} [\![Q']\!]$. Explicitly writing some of the intermediate states, we can construct the path

$$\pi_1 : [\![Q]\!] \xrightarrow{b_h} t_1 \xrightarrow{b_h} \cdots \xrightarrow{b_h} t_k \xrightarrow{\tau_m}{}^{k} [\![Q']\!] \text{ (Lemma 10)}$$

Since $\forall j \in \mathbb{N}$, $\exists n \in \mathbb{N}$ such that $[\![P]\!] \xrightarrow{\tau_l}\xrightarrow{\tau_h}{}^{n}\xrightarrow{\overline{b_h}}{}^{j}\xrightarrow{\tau_m}{}^{n+1} [\![P']\!]$ (induction hypothesis), it is also true for all j not less than k: $0 < k \leq j$. Then we can find a path

$$\pi_2 : [\![P]\!] \xrightarrow{\tau_l}\xrightarrow{\tau_h}{}^{n} r_0 \xrightarrow{\overline{b_h}} r_1 \xrightarrow{\overline{b_h}} \cdots \xrightarrow{\overline{b_h}} r_k \xrightarrow{\overline{b_h}} \cdots \xrightarrow{\overline{b_h}} r_j \xrightarrow{\tau_m}{}^{n+1} [\![P']\!]$$

Composing the beginning of π_2 with a $[\![Q]\!]$ to a new path ($[\![Q]\!]$ cannot preempt any action)

$$\pi_3 : [\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{\tau_l}\xrightarrow{\tau_h}{}^{n} r_0|[\![Q]\!].$$

Since h is the highest priority level, the b_h in π_1 and $\overline{b_h}$ in π_2 can always merge into a τ_h action; $t_k, t_{k+1}, \ldots, [\![Q']\!]$ and $r_j, r_{j+1}, \ldots, [\![P']\!]$ do not preempt τ_m (Lemmas 6, 9). As a result, π_3 can be prolonged:

$$\pi_3' : [\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{\tau_l}\xrightarrow{\tau_h}{}^{n} r_0|[\![Q]\!] \xrightarrow{\tau_h} r_1|t_1 \xrightarrow{\tau_h} r_2|t_2 \xrightarrow{\tau_h} \cdots$$
$$\xrightarrow{\tau_h} r_k|t_k \xrightarrow{\overline{b_h}} \cdots \xrightarrow{\overline{b_h}} r_j|t_k \xrightarrow{\tau_m}{}^{n+1} [\![P']\!]|t_k \xrightarrow{\tau_m}{}^{k} [\![P']\!]|[\![Q']\!]$$

π_3' can be rewritten as

$$\pi_3' : [\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{\tau_l}\xrightarrow{\tau_h}{}^{n+k} r_k|t_k \xrightarrow{\overline{b_h}}{}^{j-k} r_j|t_k \xrightarrow{\tau_m}{}^{n+1+k} [\![P']\!]|[\![Q']\!] = [\![S']\!]$$

Let $j' = j - k$, we can always construct such π_3' for all $k \leq j$ and thus for all $0 \leq j - k = j'$. Let $n' = n + k$, then we have found an n' that makes the lemma true for transitions derivated from (BRO-COMM).

- *Agent Identifier:* Suppose $S \overset{def}{=} P$, the proof is again similar to that in Lemma 7.
- *Relabelling:* Suppose $S = P[f]$, then $[\![S]\!] = [\![P]\!][f']$. $f'(\alpha) = \alpha$, if $\alpha \notin image(\varphi_{[\![]\!]})$; $\varphi_{[\![]\!]}(f(\alpha)) = f'(\varphi_{[\![]\!]}(\alpha))$, else.

  $S \xrightarrow{b!} S'$, then there exists d where $f(d) = b$, $P \xrightarrow{d!} P'$, and $S' = P'[f]$. The reason is the same as stated in Lemma 7.

  Because $P \xrightarrow{d!} P'$, according to the induction hypothesis, it is true that for all $j \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that
  $$[\![P]\!] \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n} \xrightarrow{\overline{d_h}}{}^{j} \xrightarrow{\tau_m}{}^{n+1} [\![P']\!].$$
  Applying (REL) $2(n+1)+j$ times, we have the path
  $$[\![P]\!][f'] \xrightarrow{f'(\tau_l)} \xrightarrow{f'(\tau_h)}{}^{n} \xrightarrow{f'(\overline{d_h})}{}^{j} \xrightarrow{f'(\tau_m)}{}^{n+1} [\![P']\!][f'].$$
  where
  - $f'(\tau_l) = f'(\varphi_{[\![]\!]}(\tau)) = \varphi_{[\![]\!]}(f(\tau)) = \varphi_{[\![]\!]}(\tau) = \tau_l$;
  - $f'(\tau_h) = \tau_h$, $f'(\tau_m) = \tau_m$ (τ_m, τ_h not in the image of $\varphi_{[\![]\!]}()$);
  - $f'(\overline{d_h}) = f'(\varphi_{[\![]\!]}(d!)) = \varphi_{[\![]\!]}(f(d!)) = \varphi_{[\![]\!]}(b!) = \overline{b_h}$.

  Replacing all the $f'(\cdot)$ in the path above, we have a new path:
  $$[\![P]\!][f'] = [\![P[f]]\!] = [\![S]\!] \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n} \xrightarrow{\overline{b_h}}{}^{j} \xrightarrow{\tau_m}{}^{n+1} [\![P']\!][f'] = [\![P'[f]]\!] = [\![S']\!].$$
- *Restriction:* Suppose $S = P\backslash L$, then $[\![S]\!] = [\![P]\!]\backslash\varphi_{[\![]\!]}(L)$. $S \xrightarrow{b!} S'$ implies $P \xrightarrow{b!} P'$ with $S' = P\backslash L$. $b_h, \overline{b_h}, \tau_l, \tau_m, \tau_h \notin \varphi_{[\![]\!]}(L)$ as before.

  As part of the induction hypothesis, it is true for all $j \in \mathbb{N}$, there exists $n \in \mathbb{N}$ such that
  $$[\![P]\!] \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n} \xrightarrow{\overline{b_h}}{}^{j} \xrightarrow{\tau_m}{}^{n+1} [\![P']\!].$$
  Apply (RES) $2(n+1)+j$ time, we have a new path
  $$[\![P]\!]\backslash\varphi_{[\![]\!]}(L) \xrightarrow{\tau_l} \xrightarrow{\tau_h}{}^{n} \xrightarrow{\overline{b_h}}{}^{j} \xrightarrow{\tau_m}{}^{n+1} [\![P']\!]\backslash\varphi_{[\![]\!]}(L).$$
  Since $[\![S]\!] = [\![P\backslash L]\!] = [\![P]\!]\backslash\varphi_{[\![]\!]}(L)$ and $[\![S']\!] = [\![P'\backslash L]\!] = [\![P']\!]\backslash\varphi_{[\![]\!]}(L)$ we have the proof.

□

Lemma 12 For all $S \in \mathscr{T}_{ABC}$, if $[\![S]\!] \Longrightarrow [\![S']\!]$ then $S \Longrightarrow S'$.

Proof: Again, because of reflexivity of $\Longrightarrow$, the lemma is trivially true for $S = S'$, we only need to consider the case when S is different from S', which means the length of the computation $[\![S]\!] \Longrightarrow [\![S']\!]$ is greater than 0.

To begin with, we find the first state on the computation $[\![S]\!] \Longrightarrow [\![S']\!]$ that is equivalent to an encoding of some process $S_1 \in \mathscr{T}_{ABC}$, and show that $S \Longrightarrow S_1$.

Formally, if S is an ABC process, for all computations starting from $[\![S]\!]$ such that for some $S' \in \mathscr{T}_{ABC}$,
$$[\![S]\!] \longrightarrow t_1 \longrightarrow t_2 \longrightarrow \cdots \longrightarrow t_n = [\![S']\!] \cdots$$
and $\forall i < n, \nexists T \in \mathscr{T}_{ABC} :\ t_i = [\![T]\!]$, then $S \longrightarrow S'$.

We prove this statement by structural induction. Trivially, the statement is true for $\mathbf{0}$. Suppose the statement is true for P, Q, P_i ($i \in I$, I an index set), and S is a process built upon them.

- *Prefixing:*
  - If $S = b?.P$ or $S = c.P$ for some $b? \in \mathscr{B}?$, $c \in \mathscr{H}_{CCS}$, then $[\![S]\!] = b_h.\tau_m.[\![P]\!]$ or $[\![S]\!] = c_l.[\![P]\!]$, respectively, and there is no computation with length greater than 0 starting from $[\![S]\!]$, since the only outgoing transition is not labelled with a reduction label, which means the condition does not hold and the implication is always true.
  - If $S = \tau.P$, then $[\![S]\!] = \tau_l.[\![P]\!]$, which means $t_n = t_1 = [\![P]\!]$, and $S \longrightarrow P = S'$, the statement holds.
  - If $S = b!.P$, then $[\![S]\!] = \tau_l.(\overline{b_h}.X + \tau_m.[\![P]\!])$, $X \overset{def}{=} \overline{b_h}.X + \tau_m.[\![P]\!]$. All possible computations with lengths greater than 0 are
    - $[\![S]\!] \xrightarrow{\tau_l} X$; or
    - $[\![S]\!] \xrightarrow{\tau_l} X \xrightarrow{\tau_m} [\![P]\!]$; or
    - $[\![S]\!] \xrightarrow{\tau_l} X \xrightarrow{\tau_m} [\![P]\!] \cdots$

    where X is not an encoding of any ABC process, because $X \xrightarrow{\tau_m} [\![P]\!]$, which means X can preempt actions with low priority, and it contradicts with Lemma 6. Thus $t_n = t_2 = [\![P]\!]$, and $S \longrightarrow P = S'$.
- *Choice:* Suppose $S = \sum_{i\in I} P_i$, then $[\![S]\!] = \sum_{i\in I}[\![P_i]\!]$. Without loss of generality, we can assume that the computation $[\![S]\!] \Longrightarrow [\![S']\!]$ derivated from $[\![P_j]\!] \Longrightarrow [\![S']\!]$ for some $j \in I$. As implied by induction hypothesis, $P_j \longrightarrow S'$, which in turn implies $S \longrightarrow S'$.
- *Parallel composition:* Suppose $S = P|Q$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$.

  As suggested by Lemma 6, any reduction with source state $[\![S]\!]$ must be labelled with τ_l.

  For a better analysis, we can *decompose* the transition $[\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{\tau_l} s_1$ into two transitions. A *decomposition* of a transition from a parallel composed process is the original transitions that led to this transition, following Tables 2, 4, 5, and 7.

  For example, the transition $a|\overline{a} \xrightarrow{\tau} \mathbf{0}|\mathbf{0}$ can be decomposed to $a \xrightarrow{a} \mathbf{0}$; $\overline{a} \xrightarrow{\overline{a}} \mathbf{0}$. ($\alpha$ is short for $\alpha.\mathbf{0}$.)

  For symmetry, we write the decomposition of $a|b \xrightarrow{a} \mathbf{0}|b$ as $a \xrightarrow{a} \mathbf{0}$; $b \xrightarrow{\mathbf{0}} b$. There might be more than one decomposition for a transition. For instance, $X|Y \xrightarrow{\tau} X|Y$ where $X \overset{def}{=} \tau.X$, $Y \overset{def}{=} \tau.Y$ can be decomposed into $X \xrightarrow{\tau} X$; $Y \xrightarrow{\mathbf{0}} Y$ or $X \xrightarrow{\mathbf{0}} X$; $Y \xrightarrow{\tau} Y$. A decomposition of a path is defined in the obvious way.

  On the other hand however, there must be at least one decomposition, since the inference rules for transitions related to parallel composition in the aforementioned Tables all have a non-empty set of condition(s).

  The transition $[\![S]\!] = [\![P]\!]|[\![Q]\!] \xrightarrow{\tau_l} s_1$ can be decomposed into:

  (i) $[\![P]\!] \xrightarrow{\tau_l} p_1$; $[\![Q]\!] \xrightarrow{\mathbf{0}} q_1$.

  (ii) $[\![P]\!] \xrightarrow{\mathbf{0}} p_1$; $[\![Q]\!] \xrightarrow{\tau_l} q_1$.

  (iii) $[\![P]\!] \xrightarrow{c_l} p_1$; $[\![Q]\!] \xrightarrow{\overline{c_l}} q_1$ for some $c \in \mathscr{H}_{CCS}$.

  We can see from the encoding at the beginning of this subsection, the action c_l only appears as the encoding of some handshake action c in ABC, and an encoding of some ABC process always follows c_l, for any $c \in \mathscr{H}_{ABC}$. Thus in case of (iii), we know that $P \xrightarrow{c} P'$ with $p_1 = [\![P']\!]$ for some P', and $Q \xrightarrow{\overline{c}} Q'$ with $q_1 = [\![Q']\!]$ for some Q'.

  Subsequently, $s_n = s_1 = p_1|q_1 = [\![P']\!]|[\![Q']\!] = [\![P'|Q']\!]$ for some P', Q'. $P \xrightarrow{c} P'$ and $Q \xrightarrow{\overline{c}} Q'$ implies $S = P|Q \xrightarrow{\tau} P'|Q'$.

We can see that cases (i) and (ii) are symmetric, and the proof for one case also applies for the other case. Without loss of generality, we only prove case (i).

When the transition $[\![S]\!] \xrightarrow{\tau_l} s_1$ decomposes into (i), it stems either from handshake or silent action in P, or a broadcast action. If it stems from handshake communication in P, then p_1 is the encoding of some P', since all $c_l \in \mathscr{A}_{CCS^{sg}}$ are followed by the encoding of some ABC process. In this case, $[\![S]\!] = [\![P]\!] | [\![Q]\!] \xrightarrow{\tau_l} p_1 | [\![Q]\!] = [\![P']\!] | [\![Q]\!] = [\![P'|Q]\!]$ and $S = P|Q \xrightarrow{\tau} P'|Q$ and we have the proof.

If the transition above stemmed from a silent action of P, the proof is similar.

If $[\![S]\!] \xrightarrow{\tau_l} s_1$ stems from the encoding of a broadcast action, we know that p_1 is not an encoding of any ABC process, because p_1 has as subterm $\overline{b_h}.X + \tau_m.[\![P']\!]$ for some $b, P', X \overset{def}{=} \overline{b_h}.X + \tau_m.[\![P']\!]$ and $p_1 \xrightarrow{\tau_m}$.

Since $[\![S]\!] \xrightarrow{\tau_l} s_1$ stems from a broadcast action, for $[\![S]\!]$ to reduce to a state that is an encoding of some ABC process, the computation must be in such form:

$$[\![S]\!] \xrightarrow{\tau_l} s_1 \xrightarrow{\tau_h}{}^{n} s_{n+1} \xrightarrow{\tau_m}{}^{n+1} s_{2n+2} = [\![S']\!] \text{ where } s_1 \xrightarrow{\overline{b_h}}$$

for some $n \in \mathbb{N}, b \in \mathscr{B}$ and S'.

Intuitively, the first transition labelled with τ_l indicates the beginning of a broadcast communication, it enables s_1 with an outgoing transition labelled with $\overline{b_h}$ for some $b \in \mathscr{B}$, willing to engage in handshake communication with any b_h in the system. If there is any b_h, it means that there were $b?$ in the source term S willing to synchronise with $b!$. The $\overline{b_h}$ and b_h communicate and form a transition τ_h. The number of τ_h corresponds to the number of participating $b?$ in the source term, and the number of τ_m that followed b_h.

After every b_h is consumed, τ_m preempts all low-prioritised actions, which corresponds to handshake actions or the beginning of other broadcast actions in ABC. Only after all τ_m are executed, including the one in the encoding of $b!$, will the system reach a state where it is an encoding of some ABC process S', which indicates the end of the broadcast communication, and the computation corresponds to a transition on a single broadcast channel (b) in the source term. Once a broadcast communication is initiated, which is indicated by the first τ_l in the computation, the system cannot engage in any other broadcast communication, since the encodings of all other $d! \in \mathscr{B}!$ in the system have a low-prioritised τ_l as prefix, and such τ_l is preempted by τ_h which results from the communication of b_h and $\overline{b_h}$.

In conclusion, the computation $[\![S]\!] \xrightarrow{\tau_l} s_1 \xrightarrow{\tau_h}{}^{n} s_{n+1} \xrightarrow{\tau_m}{}^{n+1} s_{2n+2} = [\![S']\!]$ ($s_1 \xrightarrow{\overline{b_h}}$) indicates that S does a broadcast communication over channel b, and reach the state S': $S \xrightarrow{b!} S'$.

The formal proof of this statement consists of multiple inductions, it is tedious and omitted.

Now we have discussed all possible situations, and the statement is therefore true when S is the parallel composition of P and Q.

- *Agent Identifier:* Suppose $S \overset{def}{=} P$, then $[\![S]\!] = P'$ where $P' \overset{def}{=} [\![P]\!]$. $[\![S]\!] \longrightarrow s_1 \longrightarrow s_2 \cdots \longrightarrow s_n = [\![S']\!]$ (s_i is not an encoding of any ABC process, where $1 \le i < n$) indicates a path $P' \longrightarrow s_1 \longrightarrow s_2 \cdots \longrightarrow s_n = [\![S']\!]$ ((RES)), which implies the existence of the path $[\![P]\!] \longrightarrow s_1 \longrightarrow s_2 \cdots \longrightarrow s_n = [\![S']\!]$, otherwise the transitions in the computation of P' above cannot derive from any SOS-rule of CCS^{sg}.

  This computation of $[\![P]\!]$ in turn implies that $P \longrightarrow S'$ (induction hypothesis), which means that $S \longrightarrow S'$ and we have the proof.

- *Relabelling:* Suppose $S = P[f]$, then $[\![S]\!] = [\![P[f]]\!] = [\![P]\!][f']$. Any computation starting from $[\![P]\!]$ where p_n of all p_i $(1 \le i \le n)$ is the only state that is an encoding of some ABC process P':

  $$[\![P]\!] \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} p_2 \cdots \xrightarrow{\alpha_n} p_n \ (\alpha_i \in \{\tau_l, \tau_m, \tau_h\})$$

  implies a new computation by applying (REL) n-times

  $$[\![P]\!][f'] \xrightarrow{f'(\alpha_1)} p_1[f'] \xrightarrow{f'(\alpha_2)} p_2[f'] \cdots \xrightarrow{f'(\alpha_n)} p_n[f']$$

  which can be rewritten as

  $$\pi_3 : [\![P[f]]\!] \xrightarrow{f'(\alpha_1)} p_1[f'] \xrightarrow{f'(\alpha_2)} p_2[f'] \cdots \xrightarrow{f'(\alpha_n)} p_n[f'] = [\![P']\!][f'] = [\![P'[f]]\!]$$

  where $P'[f] = S'$, $p_n[f']$ of all $p_i[f']$ $(1 \le i \le n)$ is the first state that is an encoding of some ABC process.

  We can prove by structural induction that $p_i[f']$ is the encoding of some $P \in \mathscr{T}_{ABC}$ implies p_i is the encoding of some $Q \in \mathscr{T}_{ABC}$. The proof is not special and omitted. Intuitively, since any relabelling function does not change priority level, or relabels τ, or turn output action into input action, the structure is preserved by applying the relabelling functions. The relabelling of low-prioritised names does not effect whether a CCS^{sg} process is an encoding of some ABC process, since any α_l $(\alpha \in \mathscr{H}_{ABC} \uplus \{\tau\})$ itself is an encoding of α, as shown in the encoding. The relabelling of broadcast names has no effect, either. If p_i contains the encoding of some broadcast action, and we change the b in $\tau_l.(\overline{b_h}.X + \tau_m.[\![P]\!])$ $(X \overset{def}{=} \overline{b_h}.X + \tau_m.[\![P]\!])$ to any broadcast name, the resulting process is still an encoding of some ABC process. The same if p_i conains the encoding of some receive action.

  As a result, $p_n[f']$ is the first state on the path such that it is an encoding of an ABC process.

  Because $P \xrightarrow{\alpha} P'$ where $\alpha \in \mathscr{B} \uplus \{\tau\}$ (induction hypothesis), by applying (REL) we have $S = P[f] \xrightarrow{f(\alpha)} P'[f] = S'$, where $f(\alpha) \in \mathscr{B} \uplus \{\tau\}$ and we have the proof.

- *Restriction:* Suppose $S = P \backslash L$, then $[\![S]\!] = [\![P]\!] \backslash \varphi_{[\![]\!]}(L)$. Because broadcast and silent actions cannot be restricted, we can easily prove this statement with $S' = P' \backslash L$ for relevant P'.

Finally, we proceed to prove the lemma. If $[\![S]\!] \Longrightarrow [\![S']\!]$ by

$$[\![S]\!] \longrightarrow s_{11} \longrightarrow s_{12} \longrightarrow \cdots \longrightarrow [\![S_1]\!] \longrightarrow s_{21} \cdots \longrightarrow [\![S_2]\!] \longrightarrow s_{31} \cdots \longrightarrow [\![S_k]\!] = [\![S']\!]$$

where s_{ij} is not the encoding of any ABC processes for all relevant i, j. Applying the result we obtained above k-times, we have $S \Longrightarrow S_1 \Longrightarrow S_2 \Longrightarrow \cdots \Longrightarrow S_k = S'$, and we have proven that if $[\![S]\!] \Longrightarrow [\![S']\!]$ then $S \Longrightarrow S'$. □

Lemma 13 If $[\![S]\!] \longmapsto^{\omega}$ then $S \longmapsto^{\omega}$ for all $S \in \mathscr{T}_{ABC}$.

Proof: By structural induction. $[\![\mathbf{0}]\!] = \mathbf{0}$ is not divergent, thus the lemma is true for $[\![\mathbf{0}]\!]$.

- *Prefixing:* Suppose $S = \alpha.P$.
  - if $\alpha \in \{\tau\}$, then $[\![S]\!] = \tau_l.[\![P]\!]$, and $[\![S]\!] \longmapsto^{\omega}$ if and only if $[\![P]\!] \longmapsto^{\omega}$, which implies $P \longmapsto^{\omega}$ (induction hypothesis), and thus $S = \tau.P \longmapsto^{\omega}$.
  - if $\alpha \in \mathscr{H}_{ABC}$, then there is no infinite computation from S, because α_l is not a reduction label.
  - if $\alpha \in \mathscr{B}?$, then $[\![S]\!] = b_h.\tau_m.[\![P]\!]$, and the proof is same as the above case.

- if $\alpha \in \mathscr{B}!$, then $[\![S]\!] = \tau_l.(\overline{b_h}.X + \tau_m.[\![P]\!]), X \overset{def}{=} \overline{b_h}.X + \tau_m.[\![P]\!]$. The only infinite computation must include
$$[\![S]\!] \xrightarrow{\tau_l} X \xrightarrow{\tau_m} [\![P]\!] \cdots$$
thus $[\![S]\!] \longmapsto^{\omega}$ if and only if $[\![P]\!] \longmapsto^{\omega}$, and $P \longmapsto^{\omega}$ implies $S \longmapsto^{\omega}$.

- *Choice:* Suppose $S = \sum_{i\in I} P_i$, the proof is straightforward.
- *Parallel composition:* Suppose $S = P|Q$, then $[\![S]\!] = [\![P]\!]|[\![Q]\!]$. When we decompose the infinite computation starting from $[\![S]\!]$, each transition is decomposed either into a silent action of $[\![P]\!]$ ($[\![Q]\!]$) and no action of $[\![Q]\!]$ ($[\![P]\!]$), or a handshake communication between $[\![P]\!]$ and $[\![Q]\!]$. Either way, for $[\![S]\!]$ to diverge, either $[\![P]\!]$ or $[\![Q]\!]$ or both are divergent, which implies P or Q or both are divergent. Since there is no preepmtion mechanism in ABC, $S = P|Q$ is also divergent.
- *Agent Identifier:* Suppose $S \overset{def}{=} P$, then $[\![S]\!] = P'$ where $P' \overset{def}{=} [\![P]\!]$.
$[\![S]\!] \longmapsto^{\omega}$ means that there is an infinite path
$$\pi_1 : [\![S]\!] = P' \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} s_3 \longrightarrow \cdots$$
where α_i is silent action for all relevant i.
Since $P' \overset{def}{=} [\![P]\!]$, there must also be an infinite path
$$\pi_2 : [\![P]\!] \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} p_2 \xrightarrow{\alpha_3} p_3 \longrightarrow \cdots$$
otherwise the transitions in π_1 cannot be derived from any SOS-rules of CCS^{sg}. This means that $[\![P]\!] \longmapsto^{\omega}$, which in turn implies $P \longmapsto^{\omega}$ and thus by applying the SOS-rule for agent identifier in Table 7 we acquire an infinite computation with S as the source state, i.e. $S \longmapsto^{\omega}$.
- *Relabelling:* Suppose $S = P[f]$, then $[\![S]\!] = [\![P[f]]\!] = [\![P]\!][f']$. $[\![S]\!] \longmapsto^{\omega}$ means that there is an infinite path:
$$\pi_1 : [\![S]\!] = [\![P]\!][f'] \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \xrightarrow{\alpha_3} s_3 \xrightarrow{\alpha_4} \cdots$$
where $\alpha_i \in \{\tau_l, \tau_m, \tau_h\}$ for all relevant i.
Since renaming functions do not map silent actions to non-silent actions, and they do not change priority levels, we always have $f(\tau_i) = \tau_i$ where $i = h$, $i = m$, or $i = l$ for any renaming function f. This means that the existence of π_1 implies the existence of π_2:
$$\pi_2 : [\![P]\!] \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} p_2 \xrightarrow{\alpha_3} p_3 \xrightarrow{\alpha_4} \cdots$$
where $\alpha_i \in \{\tau_l, \tau_m, \tau_h\}$ and $s_i = p_i[f']$ for all relevant i. This means that $[\![S]\!] \longmapsto^{\omega}$ implies $[\![P]\!] \longmapsto^{\omega}$ which in turn implies $P \longmapsto^{\omega}$ (induction hypothesis).
Because $S = P[f]$, f maps silent action to silent action, $P \longmapsto^{\omega}$ implies $S = P[f] \longmapsto^{\omega}$.
- *Restriction:* Suppose $S = P\backslash L$, then $[\![S]\!] = [\![P]\!]\backslash\varphi_{[\![]\!]}(L)$. $[\![S]\!] = [\![P]\!]\backslash\varphi_{[\![]\!]}(L)$ is divergent implies $[\![P]\!]$ is divergent, thus P is divergent and $S = P\backslash L$ is divergent, because silent actions cannot be elements of L.

□

Now we have proved operational completeness with Lemma 4, and operational soundness with Lemma 12. Divergence reflection is proved in Lemma 13. In conclusion, the encoding proposed at the beginning of this subsection is valid up to the aforementioned criteria.

3.6 CCS^{sg} to CCS, CCSS, and ABC

In their paper, Versari, Busi and Gorrieri separated CCS^{sg} from π-Calculus as well as $b\pi$-Calculus [14]. We show in the following propositions that the same result extends to CCS, CCSS, and ABC.

Proposition 11 There is no homomorphic (Criterion 2), weakly-observation-respecting (5.2) encoding from CCS^{sg} to ABC.

Proof: Consider the process P with the set of ovservables $Obs = \{n_l, y_l\}$, where

$$P = \overline{m_l} \mid \overline{s_h} \mid \overline{q_h} \mid k_h.(s_h|q_h|\overline{k_h}) \mid m_l.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\ ^{5}$$

There are only two priority levels considered here. $h = 0$ stands for high priority, $l = 1$ stands for low priority.

There is only one maximal computation of P, where the reductions originated from the handshake communication over m_l, s_h, r_l, q_h, consecutively. The subterms that form the next reduction are highlighted in bold, and we write P instead of $P|\mathbf{0}$ for readability:

$$\begin{aligned}
\mathscr{C}_1 :& \overline{\mathbf{m_l}} \mid \overline{s_h} \mid \overline{q_h} \mid k_h.(s_h|q_h|\overline{k_h}) \mid \mathbf{m_l}.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
\xrightarrow{\tau_l}& \overline{\mathbf{s_h}} \mid \overline{q_h} \mid k_h.(s_h|q_h|\overline{k_h}) \mid \mathbf{s_h}.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
\xrightarrow{\tau_h}& \overline{q_h} \mid k_h.(s_h|q_h|\overline{k_h}) \mid s_h.q_h.(\overline{k_h}|n_l) \mid \overline{\mathbf{r_l}} \mid \mathbf{r_l}.q_h.y_l\\
\xrightarrow{\tau_l}& \overline{\mathbf{q_h}} \mid k_h.(s_h|q_h|\overline{k_h}) \mid s_h.q_h.(\overline{k_h}|n_l) \mid \mathbf{q_h}.y_l\\
\xrightarrow{\tau_h}& k_h.(s_h|q_h|\overline{k_h}) \mid s_h.q_h.(\overline{k_h}|n_l) \mid y_l
\end{aligned}$$

where $Obs(\mathscr{C}_1) = \{y_l\}$.

The process $P|P$ has different computations, one of which is:

$$\begin{aligned}
\mathscr{C}_2 :& P|P \xrightarrow{\tau_l}\xrightarrow{\tau_h} \text{(handshake communication over } m_l \text{ and } s_h)\\
& \overline{q_h} \mid k_h.(s_h|q_h|\overline{k_h}) \mid \mathbf{s_h}.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l \mid\\
& \overline{m_l} \mid \overline{\mathbf{s_h}} \mid \overline{q_h} \mid k_h.(s_h|q_h|\overline{k_h}) \mid m_l.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
\xrightarrow{\tau_h}& \overline{q_h} \mid k_h.(s_h|q_h|\overline{k_h}) \mid \mathbf{q_h}.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l \mid\\
& \overline{m_l} \mid \overline{\mathbf{q_h}} \mid k_h.(s_h|q_h|\overline{k_h}) \mid m_l.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
\xrightarrow{\tau_h}& \overline{q_h} \mid \mathbf{k_h}.(s_h|q_h|\overline{k_h}) \mid \overline{\mathbf{k_h}} \mid n_l \mid \overline{r_l} \mid r_l.q_h.y_l \mid\\
& \overline{m_l} \mid k_h.(s_h|q_h|\overline{k_h}) \mid m_l.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
\xrightarrow{\tau_h}& \overline{q_h} \mid s_h \mid q_h \mid \overline{\mathbf{k_h}} \mid n_l \mid \overline{r_l} \mid r_l.q_h.y_l \mid\\
& \overline{m_l} \mid \mathbf{k_h}.(s_h|q_h|\overline{k_h}) \mid m_l.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
\xrightarrow{\tau_h}& \overline{\mathbf{q_h}} \mid s_h \mid q_h \mid n_l \mid \overline{r_l} \mid r_l.q_h.y_l \mid\\
& \overline{m_l} \mid s_h \mid \mathbf{q_h} \mid \overline{k_h} \mid m_l.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
\xrightarrow{\tau_h}& s_h \mid q_h \mid n_l \mid \overline{r_l} \mid r_l.q_h.y_l \mid\\
& \overline{m_l} \mid s_h \mid \overline{k_h} \mid m_l.s_h.(s_h.q_h.(\overline{k_h}|n_l) \mid \overline{r_l} \mid r_l.q_h.y_l)\\
& = Q
\end{aligned}$$

[5] In their original paper, Versari, Busi and Gorrieri studied a simpler variation of CCS^{sg} with slightly different syntax than in this thesis. We changed the representation so it is uniform with other sections of this thesis, but the process itself is essentially the same.

and $n_l \in Obs(\mathscr{C}_2)$.

When the computation reaches state Q, there is no more $\overline{q_h}$ in Q, which means that q_h in Q will never participate in a handshake communication, hence any computation of Q will never exhibit the observable y_l, since all y_l in Q have q_h as prefix.

When we list all maximal computations of $P|P$, they all inevitably has Q or a state that is structurally congruent (Table 8) with Q as an intermediate state. This means that for all maximal computations $\mathscr{C}_i$ of $P|P$, it holds that $Obs(\mathscr{C}_i) = \{n_l\}$ and $y_l \notin Obs(\mathscr{C}_i)$.

Suppose there is a homomorphic encoding from CCS^{sg} to ABC that is weakly observation-respecting, then the existence of the maximal computation $\mathscr{C}_1$ of P implies the existence of a maximal computation $\mathscr{C}_1'$ of $[\![P]\!]$ that satisfies $Obs(\mathscr{C}_1') = Obs(\mathscr{C}_1) = \{y_l\}$ (weak observation-respecting encoding).

This together with $[\![P|P]\!] = [\![P]\!]|[\![P]\!]$ (homomorphy) implies that there exists a maximal computation $\mathscr{C}_1''$ of $[\![P]\!]|[\![P]\!]$ such that $Obs(\mathscr{C}_1') = \{y_l\} \subseteq Obs(\mathscr{C}_1'')$ (Lemma 15).

This contradicts with the result obtained above that no maximal computation of $[\![P]\!]|[\![P]\!]$ exhibits y_l.
□

Lemma 14 Given a set of observables $Obs \subseteq \mathscr{H}_{ABC}$, for all P, $Q \in \mathscr{T}_{ABC}$, if P has a computation $\mathscr{C}$, then there exists a computation $\mathscr{C}'$ of $P|Q$ such that $Obs(\mathscr{C}) \subseteq Obs(\mathscr{C}')$.

Proof: Suppose $\mathscr{C}$ is a (possibly infinite) computation of P. If $\mathscr{C}$ contains no reduction, then $Obs(\mathscr{C}) = Obs(P)$ (Definition 4), then there is a computation $\mathscr{C}'$ of $P|Q$ with no reduction and $Obs(\mathscr{C}') = Obs(P|Q)$.

For all S, $T \in \mathscr{T}_{ABC}$ and $x \in Obs \subseteq \mathscr{H}_{ABC}$, $S \downarrow x$ implies $S|T \downarrow x$ (please refer to SOS-rules for ABC in Section 2.3), which means that $Obs(S) \subseteq Obs(S|T)$. As a result, $Obs(\mathscr{C}) = Obs(P) \subseteq Obs(P|Q) = Obs(\mathscr{C}')$.

If $\mathscr{C}$ contains at least one reduction, then we can write the (possibly infinite) computation $\mathscr{C}$ explicitly:

$$\mathscr{C} : p_0 \xrightarrow{\alpha_1} p_1 \xrightarrow{\alpha_2} p_2 \xrightarrow{\alpha_3} \cdots \longrightarrow p_n \cdots$$

where $p_0 = P$ and $\alpha_i \in \{\tau\} \cup \mathscr{B}!$, $0 \leq i$.

If $\alpha_1 = \tau$, then by applying (PAR-L) in Table 2, we acquire a reduction $P|Q = p_0|Q \xrightarrow{\alpha_1} p_1|q_1$ where $q_1 = Q$; if $\alpha_1 = b!$ for some $b \in \mathscr{B}$, then by applying (BRO-L) (if $Q \overset{b?}{\not\longrightarrow}$) or (BRO-C) (if $Q \xrightarrow{b?} Q'$) in Table 5, we again acquire a computation $P|Q \xrightarrow{\alpha_1} p_1|q_1$ where $q_1 = Q$ or $q_1 = Q'$, respectively.

We can do the same for all α_i and acquire a computation $\mathscr{C}'$:

$$\mathscr{C}' : p_0|q_0 \xrightarrow{\alpha_1} p_1|q_1 \xrightarrow{\alpha_2} p_2|q_2 \xrightarrow{\alpha_3} \cdots \longrightarrow p_n|q_n \cdots$$

where $p_0 = P$, $q_0 = Q$.

The sets of observables satisfy $Obs(\mathscr{C}) = \{x \mid \exists p_i : p_i \downarrow x (i \in \mathbb{N})\}$ and $Obs(\mathscr{C}') = \{x \mid \exists p_i|q_i : p_i|q_i \downarrow x (i \in \mathbb{N})\}$ (Definition 4). If $x \in Obs(\mathscr{C})$, then there exists p_i such that $p_i \downarrow x$ and thus $p_i|T \downarrow x$ for all T. This in turn implies that there exists $p_i|q_i$ such that $p_i|q_i \downarrow x$, and hence $x \in Obs(\mathscr{C}')$, which implies $Obs(\mathscr{C}) \subseteq Obs(\mathscr{C}')$.

The existence of $\mathscr{C}'$ proves the lemma. □

Lemma 15 Given a set of observables $Obs \subseteq \mathscr{H}_{ABC}$, for all P, $Q \in \mathscr{T}_{ABC}$, if P has a <u>maximal</u> computation $\mathscr{C}$, then there exists a <u>maximal</u> computation $\mathscr{C}'$ of $P|Q$ such that $Obs(\mathscr{C}) \subseteq Obs(\mathscr{C}')$.

Proof: If $\mathscr{C}$ is a maximal computation of P, then by definition, $\mathscr{C}$ is also a computation, which implies that there exists a computation $\mathscr{C}'$ of $P|Q$ such that $Obs(\mathscr{C}) \subseteq Obs(\mathscr{C}')$ (Lemma 14).

If $\mathscr{C}'$ is maximal, the lemma is true. If $\mathscr{C}'$ is not maximal, then we can extend $\mathscr{C}'$ to a maximal computation $\mathscr{C}''$. From the definition of Obs we can see that $Obs(\mathscr{C}') \subseteq Obs(\mathscr{C}'')$, and thus $Obs(\mathscr{C}) \subseteq Obs(\mathscr{C}'')$. □

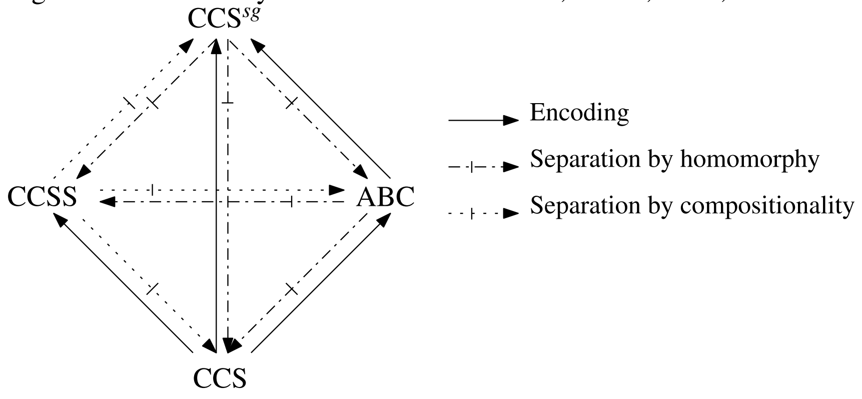
Proposition 12 There is no homomorphic (Criterion 2), weakly-observation-respecting (5.2) encoding from CCS^{sg} to CCS or CCSS.

Proof: Lemma 15 stays true if we replace ABC by CCS or CCSS ((PAR-L), (PAR-R) in Table 2), and with the same reasoning as in the proof for Proposition 11, we can prove this proposition. □

4 Conclusion

The graph in Figure 8 shows the main results of this thesis: encodability relations between CCS, CCSS, ABC, and CCS^{sg}. Each arrow represents an encodability or separation result.

Figure 8: Encodability Relations Between CCS, CCSS, ABC, and CCS^{sg}



As expected, CCS can be trivially encoded into its extensions. These encodings respect all reasonable criteria. The encoding from the extensions of CCS to CCS is, however, non-trivial. The existence of a non-compositional encoding, which is supported by the fact that CCS can model κ-bounded process graphs [6], is not satisfactory, as compositionality is usually required in expressiveness results, if not its stronger version homomorphy.

However, CCSS is separated from CCS if we require compositionality. In a compositional setting, a choice operator in CCSS must be encoded into a choice operator in CCS. As a result, we cannot encode a process in CCSS where a choice operator binds a process that does signalling into a CCS process. The emission of a signal does not change the state of the process, hence no choice is actually made. This mechanism of the signalling operator cannot be mimicked by CCS, neither by ABC or CCS^{sg}.

Another separation result is that ABC cannot be homomorphically encoded into CCS or CCSS. In ABC, once a broadcast communication of a certain channel is initiated, all parallel components listening to the same channel are forced to synchronise. The broadcast communication is summarised as one action, no matter how many parallel components participated. In CCS and CCSS, the parallel composition is monotonic [1]. While homomorphically encoding a broadcast communication, we inevitably end up with a process that does more barb than intended. The broadcast mechanism and the requirement for a homomorphic encoding separates ABC from CCS and CCSS.

This result, however, does not extend to CCS^{sg}. In CCS^{sg}, due to different levels of priority assigned to a process, we can model a broadcast communication by preempting all other actions until the broadcast is finalised and all participating parallel components have transformed into their next states. We presented one such encoding and proved its correctness up to a set of expressiveness criteria.

A homomorphic encoding of the backwards direction (CCS^{sg} to ABC) is, however, not existent. Versari, Busi and Gorrieri separated CCS^{sg} from π-Calculus and $b\pi$-Calculus [14]. The same process they examined as counterexample is used in this thesis to separate CCS^{sg} from ABC. In CCS^{sg}, we can model a process where it alone can only barb "yes" and this process in parallel with itself can only barb "no". This behaviour is not preserved when we encode this process to ABC homomorphically. When a parallel component in an ABC process barbs "yes", we cannot prevent the whole process to also barb "yes", which means that we have introduced unwanted behaviour which is not observed in the original ABC process.

This statement concerning parallel composition is also true for CCS and CCSS, and thus the separation result extends to CCS and CCSS.

In conclusion, the extensions of CCS are linked to CCS by encodability results on both directions, but the extensions can be separated from CCS when we require homomorphy or compositionality, respectively; ABC can be encoded into CCS^{sg}; CCSS is incomparable with both ABC and CCS^{sg}.

Strengthening the separation results in this thesis is planned as future work. Homomorphy is strong enough to separate priority, broadcast, and signal, but we would like to weaken it into compositionality and gain a stronger separation result, or if it is impossible, give compositional encodings between the aforementioned calculi. Also, weak bisimulation and observation-respecting encodings are required in various works [5, 9, 12, 14], but they are still rather strong, and we plan to weaken it, or yield some reasoning that any weakening would break the result.

References

[1] J. Åman Pohjola & J. Parrow (2016): *The Expressive Power of Monotonic Parallel Composition.* 9632, pp. 780–803, doi:10.1007/978-3-662-49498-1_30.

[2] M. Bouwman (2018): *Liveness analysis in process algebras.* Available at `https://www.win.tue.nl/~timw/downloads/bouwman_seminar.pdf`.

[3] R. Cleaveland, G. Lüttgen & V. Natarajan (2001): *CHAPTER 12 - Priority in Process Algebra.* In J. Bergstra, A. Ponse & S. Smolka, editors: Handbook of Process Algebra, Elsevier Science, Amsterdam, pp. 711 – 765, doi:https://doi.org/10.1016/B978-044482830-9/50030-8.

[4] V. Dyseryn, R. van Glabbeek & P. Höfner (2017): *Analysing Mutual Exclusion using Process Algebra with Signals.* Electronic Proceedings in Theoretical Computer Science 255, p. 18–34, doi:10.4204/eptcs.255.2.

[5] C. Ene & T. Muntean (1999): *Expressiveness of point-to-point versus broadcast communications.* In G. Ciobanu & G. Păun, editors: Fundamentals of Computation Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 258–268.

[6] R.J. van Glabbeek (1995): *On the Expressiveness of ACP.* In A. Ponse, C. Verhoef & S.F.M. van Vlijmen, editors: Algebra of Communicating Processes, Springer London, London, pp. 188–217.

[7] R.J. van Glabbeek & P. Höfner (2015): *Progress, Fairness and Justness in Process Algebra.* CoRR abs/1501.03268. Available at `http://arxiv.org/abs/1501.03268`.

[8] D. Gorla (2010): *Towards a Unified Approach to Encodability and Separation Results for Process Calculi.* Inf. Comput. 208(9), pp. 1031–1053, doi:10.1016/j.ic.2010.05.002.

[9] C.T. Jensen (1994): *Interpreting broadcast communication in ccs with priority choice.* In: Proceedings of the 6th Nordic Workshop on Programming Theory, Citeseer, pp. 49–70.

[10] R. Milner (1980): *A Calculus of Communicating Systems.* Springer-Verlag, Berlin, Heidelberg.

[11] J. Parrow (2008): *Expressiveness of Process Algebras.* Electron. Notes Theor. Comput. Sci. 209, pp. 173–186, doi:10.1016/j.entcs.2008.04.011.

[12] I. Phillips (2008): *CCS with priority guards*. *The Journal of Logic and Algebraic Programming* 75(1), pp. 139 – 165, doi:https://doi.org/10.1016/j.jlap.2007.06.005. Algebraic Process Calculi. The First Twenty Five Years and Beyond. III.

[13] K.V.S. Prasad (1991): *A calculus of broadcasting systems*. In S. Abramsky & T.S.E. Maibaum, editors: *TAPSOFT '91*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 338–358.

[14] C. Versari, N. Busi & R. Gorrieri (2009): *An expressiveness study of priority in process calculi*. *Mathematical Structures in Computer Science* 19(6), p. 1161–1189, doi:10.1017/S0960129509990168.