An Encoding of the CSP External Choice operator to ACP_τ

Leon Lee

January 16, 2025

1.1 Prerequisites

We represent a language $\mathscr{L}$ as a pair $(\mathbb{P}, [\![\ ]\!])$, where $\mathbb{P}$ is a set of valid expressions in $\mathscr{L}$, and $[\![\ ]\!]$ is a mapping $[\![\ ]\!] : \mathbb{P} \to \mathscr{D}$ from $\mathbb{P}$ to a set of meanings $\mathscr{D}$. We also define $A \subseteq \mathbb{P}$, where A is the set of actions

Definition 1.1.1: Subsets of A

We define some subsets of A which we will use in our encodings

- $A_0 \subseteq A$ is the set of actions that actually get used in processes
- $H_0 = A - A_0$ is the set of working space operators, or any other action that doesn't get used
- $H_1 = A_0 \uplus \{\texttt{first}, \texttt{next}, \texttt{choose}\}$ is the set of actions, plus some working operators

In general, $A_0 \subseteq H_1 \subseteq A$.
Note that the silent step is not defined in A, and we will define A_τ to be $A \cup \{\tau\}$

Working similarly to [3], we will work in the language ACP_τ, with an additional operator called "Functional Renaming". I will refer to this language as ACP_F^τ. To define translations from CSP to ACP_F^τ we utilise two operators - $\rho_F(p)$ and ∂_H.

- **Functional Renaming**: $\rho_F(p)$ takes a function $F : A \to A$, and where applicable, applies F to each action in p
- **Encapsulation**: ∂_H takes a subset $H \subseteq A$, and restricts any actions that are in H

1.2 Triggering in ACP

We define an operator $\Gamma(p)$ which acts like the MEIJE triggering operator on a process $p \in \mathbb{P}$. For this, we will use the notation of a^∞ to mean $a.a.a\ldots$. First, we define a function f_1 and communications for the operations `first` and `next`.

Definition 1.2.1: F1

Define functions $f_1 : A \to A$ and $f_2 : A \to A$ where

$$f_1(a_{\texttt{first}}) = a_{\texttt{first}} \quad f_2(a_{\texttt{first}}) = a_{\texttt{ini}}$$
$$f_1(a_{\texttt{next}}) = a \quad f_2(a_{\texttt{next}}) = a$$

Definition 1.2.2: Communications

Define communications where

$$a|\texttt{first} = a_{\texttt{first}}$$
$$a|\texttt{next} = a_{\texttt{next}}$$

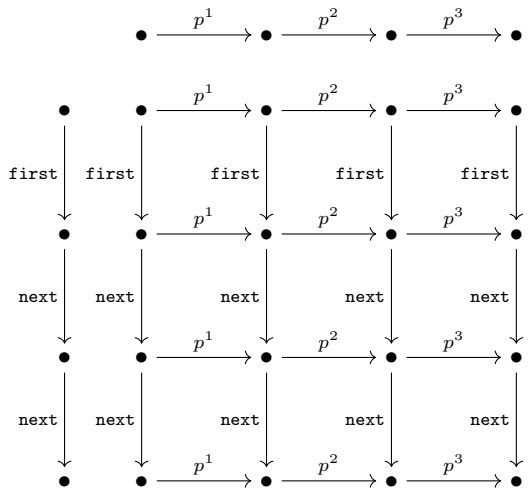
We can now define $\Gamma(p)$ as such:

Definition 1.2.3: Triggering in ACP

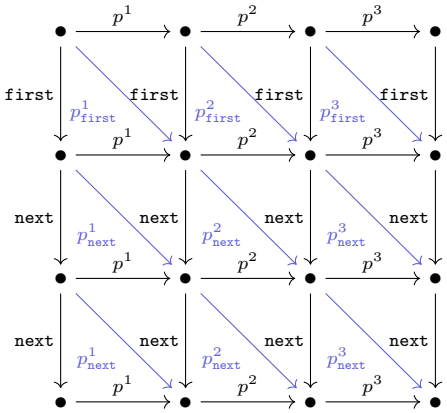
$$\Gamma(p) := f_2(\rho_{f_1}[\partial_{H_1}(p||\texttt{first}(\texttt{next}^\infty))])$$

is an operator that turns a trace of a process p, $p_1.p_2.p_3\ldots$ into the trace

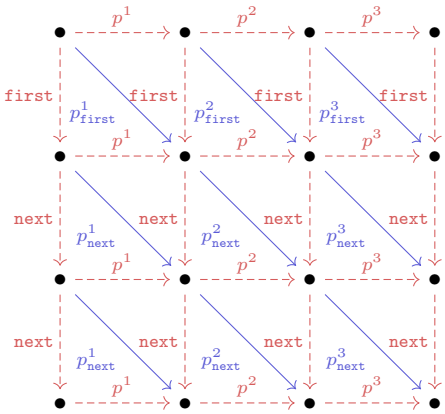
$$p^1_{\texttt{ini}}.p^2.p^3.p^4\ldots.$$



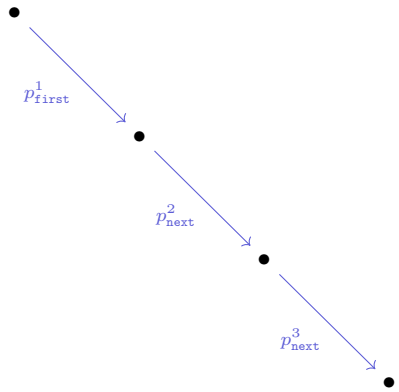
(a) Step 1 - Left Merge on both sides



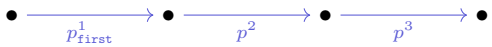
(b) Step 1 - Communications



(c) Step 2 - Restriction of H_1



(d) Step 2 - Result of Restriction



(e) Result of Functional Renaming

Figure 1: Example of $\Gamma(p)$ applied to $p = p^1.p^2.p^3$

This works in the following method:

1. Merge the process p with the process $\texttt{first}.\texttt{next}.\texttt{next}\ldots$. Via 1.2.2, this will produce a lattice of p and $\texttt{first}.(\texttt{next}^\infty)$, with communications on every square, but most importantly, a chain of communications going down the centre of the form.

$$p^1_{\texttt{first}}.p^2_{\texttt{next}}.p^3_{\texttt{next}}\cdots \tag{1}$$

2. Restrict the actions in H_1. Since both $p, \texttt{first}.(\texttt{next}^\infty) \in H_1$ this effectively restricts both sides of the left merge, leaving only communications from the initial state. This leaves equation 1 as the only remaining trace.

3. Apply ρ_{f_1} to equation 1. Via 1.2.1, the final result is

$$p^1_{\texttt{first}}.p^2.p^3\dots \tag{2}$$

Which is exactly as stated in Definition 1.2.3.

Note that since $\tau \notin A$, ∂_{H_1} will not restrict τ, and additionally since τ does not communicate, Step 2 effectively becomes any amount of τ steps followed by the diagonal trace immediately following that. This results in cases $\Gamma(p)$ where $p = \tau.p^1.p^2\dots$ becoming the trace

$$\tau.p^1_{\texttt{first}}.p^2.p^3\dots$$

effectively skipping τ's, then acting the same as processes that don't start with a τ.

1.3 A Translation of CSP external choice

The external choice operator $\Box$ is defined with the following rules:

$$\frac{P \xrightarrow{a} P'}{P\Box Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{a} Q'}{P\Box Q \xrightarrow{a} Q'} \qquad \frac{P \xrightarrow{\tau} P'}{P\Box Q \xrightarrow{\tau} P'\Box Q} \qquad \frac{Q \xrightarrow{\tau} Q'}{P\Box Q \xrightarrow{a} P\Box Q'} \tag{3}$$

In other words, we can take an external choice by the user, and additionally an internal action will still let an external choice be made after the internal move has been made. This differs from the ACP Alternative Choice operator (+), as + will not let you select externally if an internal action is made.

For our encoding, we take $H_1 = A_0 \uplus \{\texttt{first}, \texttt{next}, \texttt{choose}\}$ as defined in Definition 1.1.1. We then modify Definition 1.2.2 to include an additional communication for `choose` to make

Definition 1.3.1: Communication for CSP External Choice

$$a|\texttt{first} = a_{\texttt{first}} \qquad a|\texttt{next} = a_{\texttt{next}} \qquad a_{\texttt{ini}}|\texttt{choose} = a$$

We can then define an encoding of the CSP external choice operator $\Box$ in ACP in the following equation

$$\mathscr{T}(P\Box Q) = \partial_{H_0}\Big(f_1\Big[\Gamma[\mathscr{T}(P)] \,||\, \texttt{choose} \,||\, \Gamma[\mathscr{T}(Q)]\Big]\Big)$$

On a process without τ, the function

$$\partial_{H_0}\Big(f_1\Big[\Gamma(P) \,||\, \texttt{choose} \,||\, \Gamma(Q)\Big]\Big)$$

has identical behaviour to $P + Q$

On processes with internal actions, which we will call $\mathcal{P} = \tau_P.a.P$ and $\mathcal{Q} = \tau_Q.b.Q$, where τ_P indicates the number of starting τ actions in the process, possibly 0, the τ cannot communicate with `choose` so `choose` will only communicate with a (Since only a will be labelled with `first`). However, the first action of Q will still have the name $b_{\texttt{first}}$ for $b \in A$. This lets the function effectively skip the τ, then perform `choose` on $a.P \,||\, b.Q$ which matches the behaviour of CSP $\Box$.

However, this translation is not strongly bisimilar. This is due to the final restriction ∂_{H_0}, which will usually restrict any stray $a_{\texttt{first}}$ actions (and subsequent actions/processes) that try to communicate. However, since the silent step τ is not in A, we will get left with stray τ actions. This doesn't hold for two processes $a.P$ and $\tau.Q$, since the translation would yield the process

$$a.(\tau||P) + \tau.(P + Q) \neq a.P + \tau.(P + Q)$$

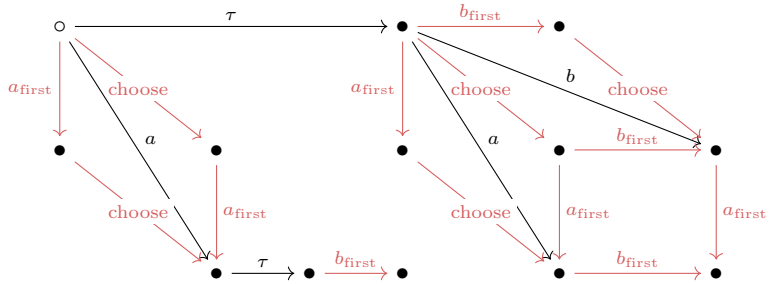


Figure 2: Counterexample for Strong Bisimilarity with the processes $P = a$ and $Q = \tau.b$. The result of the translation is $a.\tau + \tau.(a + b) \not\approx a + \tau.(a + b)$. Restricted actions are marked in red.

1.4 Proving our translation of External Choice is valid up to Rooted Branching Bisimulation

We define formally the definition of Rooted Branching Bisimulation via [1]:

Definition 1.4.1: Rooted Branching Bisimilarity

Let P and Q be two processes, and R be a relation between nodes of P and nodes of Q. R is a **Branching Bisimulation** between P and Q if:

1. The roots of P and Q are related by R
2. If $s \xrightarrow{a} s'$ for $a \in A \cup \{\tau\}$ is an edge in P, and sRt, then either
   a) $a = \tau$ and $s'Rt$
   b) $\exists t \Rightarrow t_1 \xrightarrow{a} t'$ such that sRt_1 and sRt'
3. If $s \downarrow$ and sRt then there exists a path $t \Rightarrow t'$ in Q to a node t' with $t' \downarrow$ and sRt'

4, 5 : As in 2, 3, with the roles of P and Q interchanged

R is called a **Rooted Branching Bisimulation** if the following root condition is satisfied:

- If $\text{root}(P) \xrightarrow{a} s'$ for $a \in A \cup \{\tau\}$, then there is a t' with $\text{root}(Q) \xrightarrow{a} t'$ and $s'Rt'$
- If $\text{root}(Q) \xrightarrow{a} t'$ for $a \in A \cup \{\tau\}$, then there is a s' with $\text{root}(P) \xrightarrow{a} s'$ and $s'Rt'$
- $\text{root}(g) \downarrow$ iff $\text{root}(h) \downarrow$

In other words, two processes are Rooted Branching Bisimilar if it is strongly bisimilar for the first step, and branching bisimilar for the remaining ones. This is a more desirable outcome because RBB is a congruence [2].

We can now work towards a proof that the external choice is Rooted Branching Bisimilar. In the case that P and Q don't involve an internal action τ, the translation is strongly bisimilar, and therefore RBB since Strong bisimilarity is a finer congruence than Rooted Branching Bisimilarity.

Definition 1.4.2: The translation

$$\mathscr{T}(P\Box Q) = \partial_{H_0}\Big(f_1\Big[\Gamma[\mathscr{T}(P)] \,||\, \texttt{choose} \,||\, \Gamma[\mathscr{T}(Q)]\Big]\Big)$$

Definition 1.4.3: Communications

From 1.3.1:

$$a|\texttt{first} = a_{\texttt{first}} \qquad a|\texttt{next} = a_{\texttt{next}} \qquad a_{\texttt{ini}}|\texttt{choose} = a$$

Let $P, Q \in \text{CSP}$ be two processes. WTS $P\Box Q =_{\text{RBB}} \mathscr{T}(P\Box Q)$. We want to show that any move will result in a process that satisfies RBB.

Lemma 1.4.4

For processes $P, P' \in \text{CSP}$, if $P \xrightarrow{\tau} P'$ then $\Gamma[\mathscr{T}(P)] \xrightarrow{\tau} \Gamma[\mathscr{T}(P')]$. Alternatively,

$$\Gamma[\mathscr{T}(P)] = \tau.\Gamma[\mathscr{T}(P')]$$

Proof. Something about how τ's do not get affected by the gamma function. □

Lemma 1.4.5

For processes $P, P' \in \text{CSP}$, if $P \xrightarrow{a} P'$ then for any action $a \in A_0$ we have

$$(\Gamma[\mathscr{T}(P)] \,||\, \text{choose}) \xrightarrow{a} \mathscr{T}(P') \quad \text{and} \quad (\text{choose} \,||\, (\Gamma[\mathscr{T}(P)]) \xrightarrow{a} \mathscr{T}(P')$$

and

Proof. Directly follows from communications □

Lemma 1.4.6

For processes $a.P \in \text{CSP}$ where $a \in A_0$, if there exists a process $b.P'$ where $b \in A_0$ we have

$$\partial_{H_0}(b.P' \,||\, \Gamma[\mathscr{T}(a.P)]) = \mathscr{T}(b.P')$$

Proof. The Γ function turns a function into a process equation of the form

$$a_{\texttt{ini}}.b.c\cdots$$

the process $a_{\texttt{ini}}$ does not communicate with anything other than `choose` which is not in A_0. Therefore, the restriction operator will remove all the Γ left merges, leaving $b.P'$ □

Proof. □

Lemma 1.4.7

For processes $\Rightarrow a.P \in \text{CSP}$, where $\Rightarrow$ indicates a chain of τ, possibly 0, and $a \in A_0$, if there exists a process $b.P'$ where $b \in A_0$ we have

$$\partial_{H_0}(P' \,||\, \Rightarrow \Gamma[\mathscr{T}(a.P)]) = (\mathscr{T}(P') \,||\, \Rightarrow)$$

This resulting process is Branching Bisimilar to $\mathscr{T}(a.P')$

Option 1: a action
Let $a \in A_0$ and $P' \in \text{CSP}$ s.t. $P \xrightarrow{a} P'$. In the domain of CSP, this results in the process

$$P \Box Q \xrightarrow{a} P'$$

Via Lemma 1.4.5, we can now derive the following equation

$$\mathscr{T}(P \Box Q) = \partial_{H_0}\Big(f_1\Big[\Gamma[\mathscr{T}(P)] \,||\, \texttt{choose} \,||\, \Gamma[\mathscr{T}(Q)]\Big]\Big) \xrightarrow{a} \partial_{H_0}\Big(f_1\Big[\mathscr{T}(P') \,||\, \Gamma[\mathscr{T}(Q)]\Big]\Big)$$

1. **Case 1**: $\exists b \in A_0$ and $Q' \in \text{CSP}$ s.t. $Q \xrightarrow{b} Q'$. Then via Lemma 1.4.6, this results in

   $$\mathscr{T}(P \Box Q) = \partial_{H_0}(f_1[\mathscr{T}(P')]) = \mathscr{T}(P')$$

   Which is strongly bisimilar.

2. **Case 2**: $Q \xrightarrow{\tau} Q'$. Then via Lemma 1.4.4, we can now derive the following equation:

   $$\mathscr{T}(P \Box Q) = \partial_{H_0}\Big(f_1\Big[\mathscr{T}(P') \,||\, \tau.\Gamma[\mathscr{T}(Q')]\Big]\Big)$$

   Case 2 can be repeated as many times as needed until Q reaches Case 1. Via Lemma 1.4.7, this results in

   $$\mathscr{T}(P \Box Q) = \partial_{H_0}(f_1[\mathscr{T}(P') \,||\, \Rightarrow]) = (\mathscr{T}(P') \,||\, \Rightarrow)$$

   Which is Branching Bisimilar via Lemma 1.4.7

Option 2: τ action. These results can be gained from reversing the order of P and Q using the same logic as above.

References

[1] J. C. M. Baeten and W. P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, 1990.

[2] Wan Fokkink. Rooted Branching Bisimulation as a Congruence. *Journal of Computer and System Sciences*, 60(1):13–37, February 2000.

[3] Rob van Glabbeek. On the Expressiveness of ACP. In A. Ponse, C. Verhoef, and S. F. M. van Vlijmen, editors, *Algebra of Communicating Processes*, pages 188–217, London, 1995. Springer London.