

# An Encoding of the CSP External Choice operator to $ACP_\tau$

Leon Lee

February 5, 2025

## 1.1 Prerequisites

We represent a language  $\mathcal{L}$  as a pair  $(\mathbb{T}, \llbracket \cdot \rrbracket)$ , where  $\mathbb{T}$  is a set of valid expressions in  $\mathcal{L}$ , and  $\llbracket \cdot \rrbracket$  is a mapping  $\llbracket \cdot \rrbracket : \mathbb{T} \rightarrow \mathcal{D}$  from  $\mathbb{T}$  to a set of meanings  $\mathcal{D}$ . We also define  $A \subseteq \mathbb{T}$ , 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
- $H_1 = A_0 \uplus \{\mathbf{first}, \mathbf{next}, \mathbf{choose}\}$  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  $A_\tau$  to be  $A \cup \{\tau\}$

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

- **Functional Renaming:**  $\rho_F(p)$  takes a function  $F : A \rightarrow A$ , and where applicable, applies  $F$  to each action in  $p$
- **Encapsulation:**  $\partial_H$  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  $a^\infty$  to mean  $\langle X \mid X = a.X \rangle$ . First, we define a function  $f_1$  and communications for the operations **first** and **next**.

### Definition 1.2.1: F1

Define functions  $f_1 : A \rightarrow A$  where

$$\begin{aligned} f_1(a_{\text{first}}) &= a_{\text{first}} \\ f_1(a_{\text{next}}) &= a \end{aligned}$$

### Definition 1.2.2: Communications

Define communications where

$$\begin{aligned} a|\text{first} &= a_{\text{first}} \\ a|\text{next} &= a_{\text{next}} \end{aligned}$$

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

### Definition 1.2.3: Triggering in ACP

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

is an operator that turns a trace of a process  $P$ ,  $a.b.c.\dots$  into the trace

$$a_{\text{first}}.b.c.\dots$$

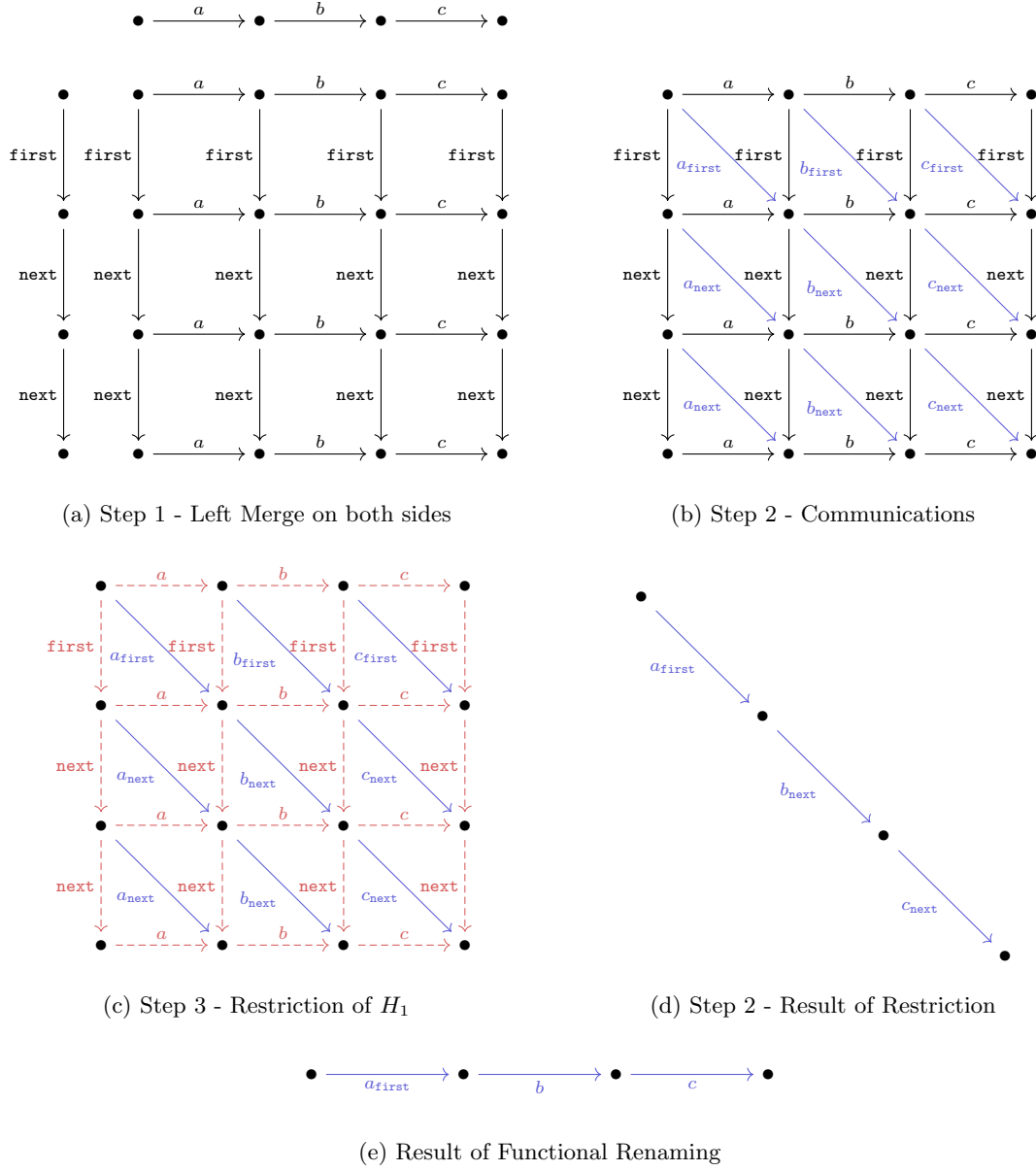


Figure 1: Example of  $\Gamma(P)$  applied to  $P = a.b.c$

This works in the following method:

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

$$a_{\text{first}}.b_{\text{next}}.c_{\text{next}} \dots \quad (1)$$

2. Restrict the actions in  $H_1$ . Since all the actions in  $P, \text{first}.\text{next}^\infty \in H_1$  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  $\rho_{f_1}$  to equation 1. Via 1.2.1, the final result is

$$a_{\text{first}}.b.c \dots \quad (2)$$

The process is now exactly as stated in Definition 1.2.3.

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

$$\tau.b_{\text{first}}.c \dots$$

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

### 1.3 A Translation of CSP external choice

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

$$\frac{P \xrightarrow{a} P'}{P \square Q \xrightarrow{a} P'} \quad \frac{Q \xrightarrow{a} Q'}{P \square Q \xrightarrow{a} Q'} \quad \frac{P \xrightarrow{\tau} P'}{P \square Q \xrightarrow{\tau} P' \square Q} \quad \frac{Q \xrightarrow{\tau} Q'}{P \square Q \xrightarrow{\tau} P \square Q'} \quad (3)$$

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  $H_1 = A_0 \uplus \{\text{first}, \text{next}, \text{choose}\}$  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

In addition to the communications defined in 1.2.2,

$$a|\text{first} = a_{\text{first}} \quad a|\text{next} = a_{\text{next}}$$

We define an additional communication for the operator **choose**

$$a_{\text{first}}|\text{choose} = a_{\text{post}}$$

#### Def 1.3.2: Compatability Func.

In addition to the function  $f_1$  defined in 1.2.1,

$$f_1(a_{\text{first}}) = a_{\text{first}} \quad f_1(a_{\text{next}}) = a$$

We define a compatibility function to prevent issues with transitivity. Let  $f_2 : A \rightarrow A$  where

$$f_2(a_{\text{post}}) = a \quad f_2(\alpha) = \alpha$$

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

$$\mathcal{T}(P \square Q) = \partial_{H_0} \left( \rho_{f_2} \left[ \Gamma[\mathcal{T}(P)] \parallel \text{choose} \parallel \Gamma[\mathcal{T}(Q)] \right] \right)$$

On a process without  $\tau$ , the function

$$\partial_{H_0} \left( \rho_{f_2} \left[ \Gamma(P) \parallel \text{choose} \parallel \Gamma(Q) \right] \right)$$

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  $\tau_P$  indicates the number of starting  $\tau$  actions in the process, possibly 0, the  $\tau$  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  $b_{\text{first}}$  for  $b \in A$ . This lets the function effectively skip the  $\tau$ , then perform **choose** on  $a.P \parallel b.Q$  which matches the behaviour of CSP  $\square$ .

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

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

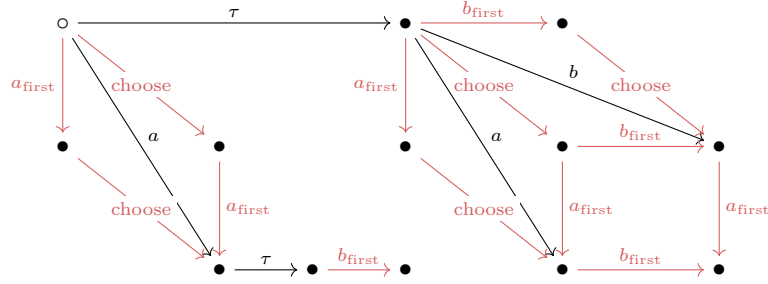


Figure 2: Counterexample for Strong Bisimilarity with the processes  $P = a$  and  $Q = \tau.b$ . The result of the translation is  $a.\tau + \tau.(a + b) \not\equiv a + \tau.(a + b)$ . Restricted actions are marked in red.

## 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.

**Definition 1.4.2: The translation**

$$\mathcal{T}(P \square Q) = \partial_{H_0} \left( f_1 \left[ \Gamma[\mathcal{T}(P)] \parallel \text{choose} \parallel \Gamma[\mathcal{T}(Q)] \right] \right)$$

**Definition 1.4.3: Communications**

From 1.3.1:

$$a|_{\text{first}} = a_{\text{first}} \quad a|_{\text{next}} = a_{\text{next}} \quad a_{\text{ini}}|_{\text{choose}} = a$$

**Lemma 1.4.4**

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

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

*Proof.* Something about how  $\tau$ 's do not get affected by the gamma function.  $\square$

**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

$$(\Gamma[\mathcal{T}(P)] \parallel \text{choose}) \xrightarrow{a} \mathcal{T}(P') \quad \text{and} \quad (\text{choose} \parallel (\Gamma[\mathcal{T}(P)])) \xrightarrow{a} \mathcal{T}(P')$$

and

*Proof.* Directly follows from communications  $\square$

**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

$$\partial_{H_0}(b.P' \parallel \Gamma[\mathcal{T}(a.P)]) = \mathcal{T}(b.P')$$

*Proof.* The  $\Gamma$  function turns a process into a process equation of the form

$$a_{\text{ini}}.b.c \dots$$

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

**Lemma 1.4.7**

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

$$\partial_{H_0}(P' \parallel \Rightarrow \Gamma[\mathcal{T}(a.P)]) = (\mathcal{T}(P') \parallel \Rightarrow)$$

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

*Proof.* using branching bisimilar axiom to some regard? or just compare to def 1.4.1

□



## 1.5 Proof of Rooted Branching Bisimilarity

*Proof.* Let  $P, Q \in \text{CSP}$  be two processes. WTS  $P \sqcap Q =_{\text{RBB}} \mathcal{T}(P \sqcap Q)$ . I.e.: we want to show that any move will result in a process that satisfies RBB.

---

**Option 1:**  $a$  action on  $P$ . 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

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

Via Lemma 1.4.5, we can now derive the following equation

$$\mathcal{T}(P \sqcap Q) = \partial_{H_0} \left( f_1 \left[ \Gamma[\mathcal{T}(P)] \parallel \text{choose} \parallel \Gamma[\mathcal{T}(Q)] \right] \right) \xrightarrow{a} \partial_{H_0} \left( f_1 \left[ \mathcal{T}(P') \parallel \Gamma[\mathcal{T}(Q)] \right] \right)$$

This is not yet a process that is comparable to  $\mathcal{T}(P')$ , so we look at the next step.

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

$$\mathcal{T}(P') = \partial_{H_0}(f_1[\mathcal{T}(P')]) = \mathcal{T}(P')$$

Which is strongly bisimilar.

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

$$\mathcal{T}(P') = \partial_{H_0} \left( f_1 \left[ \mathcal{T}(P') \parallel \tau. \Gamma[\mathcal{T}(Q')] \right] \right)$$

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

$$\mathcal{T}(P') = \partial_{H_0}(f_1[\mathcal{T}(P') \parallel \Rightarrow]) = (\mathcal{T}(P') \parallel \Rightarrow)$$

Which is Branching Bisimilar via Lemma 1.4.7

The first action ( $a$ ) can happen Strongly on the root, and any further actions are either also Strongly Bisimilar, or Branching Bisimilar. Therefore, an  $a$  action on  $P$  is Rooted Branching Bisimilar

---

**Option 2:**  $\tau$  action on  $P$ . Let  $P' \in \text{CSP}$  s.t.  $P \xrightarrow{\tau} P'$ . In the domain of CSP, this results in the process

$$P \sqcap Q \xrightarrow{\tau} P' \sqcap Q$$

Via Lemma 1.4.4, we can now derive the following equation

$$\partial_{H_0} \left( f_1 \left[ \Gamma[\mathcal{T}(P)] \parallel \text{choose} \parallel \Gamma[\mathcal{T}(Q)] \right] \right) \xrightarrow{\tau} \partial_{H_0} \left( f_1 \left[ \Gamma[\mathcal{T}(P')] \parallel \text{choose} \parallel \Gamma[\mathcal{T}(Q)] \right] \right)$$

Which is strongly bisimilar to  $\mathcal{T}(P' \sqcap Q)$ , therefore a  $\tau$  action on  $P$  is also Rooted Branching Bisimilar.

---

**Option 3, Option 4:** The same logic from option 1 and option 2 can be applied to  $Q$  in reverse to obtain processes that satisfy Rooted Branching Bisimilarity.

---

We have now exhausted cases, and therefore can conclude that our translation of CSP External Choice is Rooted Branching Bisimilar, and therefore a Congruence.  $\square$

## 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.