Encoding of CSP Square operator to ACP$_\tau$

Leon Lee

November 11, 2024

1.1 Prerequisites

Definition 1.1.1: Sets of A

- A is the set of actions
- $A_0 \subseteq A$ 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
- $H_1 = A_0 \uplus \{\text{first}, \text{second}, \text{stop}, \text{skip}, \text{next}\}$ is the set of actions, plus some working operators

In general, $H_1 \subseteq A_0 \subseteq A$, and $H_0 \subseteq A$

Definition 1.1.2: F1

Define a function $f_1 : A \to A$ where

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

Definition 1.1.3: Communication

Define communications where

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

Definition 1.1.4: Triggering in ACP

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

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

$$p_{\text{first}}.p_2.p_3.p_4.\ldots$$

1.2 A note about communication with Tau

From ACP$_\tau$ 1980, a selected list of axioms for ACP$_\tau$ are as follows:

- **CM1**: $x||y = x \mathbin{⌊\!⌊} y + y \mathbin{⌊\!⌊} x + x|y$
- **CM3**: $ax \mathbin{⌊\!⌊} y = a(x||y)$
- **TM2**: $\tau x \mathbin{⌊\!⌊} y = \tau(x||y)$
- **TC3**, **TC4**: $\tau x|y = x|\tau y = x|y$
- **T2**: $\tau x + x = \tau x$

Therefore, a trace of the form $P = \tau.\Phi$ where $\Phi = a.b.c\dots$ will have the following expansion:

$$\begin{aligned} P||\text{first}(\text{next}^\infty) &= \tau.\Phi \mathbin{⌊\!⌊} \text{first}(\text{next}^\infty) + \text{first}(\text{next}^\infty) \mathbin{⌊\!⌊} \tau.\Phi + \tau.\Phi|\text{first}(\text{next}^\infty) \\ &= \overbrace{\tau.(\Phi||\text{first}(\text{next}^\infty))}^{\textbf{TM2}} + \overbrace{\text{first}.(\text{next}^\infty||\tau.\Phi)}^{\textbf{CM3}} + \overbrace{\Phi|\text{first}(\text{next}^\infty)}^{\textbf{TC3}} \\ &= \tau.(\Phi||\text{first}(\text{next}^\infty)) + \text{first}.(\text{next}^\infty||\tau.\Phi) + a_{\text{first}}.b_{\text{next}}.c_{\text{next}}\dots \end{aligned}$$

Therefore,

$$\begin{aligned} \Gamma_P &= \rho_{f_1}[\partial_{H_1}(\tau.(\Phi||\text{first}(\text{next}^\infty) + \text{first}.(\text{next}^\infty||\tau.\Phi) + a_{\text{first}}.b_{\text{next}}\dots)] \\ &= \rho_{f_1}[\partial_{H_1}(\tau.(a_{\text{first}}.b_{\text{next}}\dots) + \text{first}.(\text{next}^\infty||\tau.\Phi) + a_{\text{first}}.b_{\text{next}}\dots)] \\ \textbf{T2} \quad &= \rho_{f_1}[\partial_{H_1}(\tau.(a_{\text{first}}.b_{\text{next}}\dots) + \overbrace{\text{first}.(\text{next}^\infty||\tau.\Phi)}^{\text{Restricted by } \partial_{H_1}})] \\ &= \rho_{f_1}[\tau.a_{\text{first}}.b_{\text{next}}.c_{\text{next}}\dots] \\ &= \tau.a_{\text{first}}.b.c\dots \end{aligned}$$

In general,

Theorem 1.2.1: Tau Escapism of Parallel processes

For a parallel process

$$p||T$$

If p is of the form $\tau.\Phi$, and T is of the form $t_1.t_2\dots$ where $t_n \in H_0$ for all n, then

$$\tau.\Phi||T \Leftrightarrow \tau.(\Phi||T)$$

In particular, for any applications of the gamma function Γ_p, if p is of the form $\tau.\Phi$, then

$$\Gamma_p \Leftrightarrow \tau.(\Gamma_\Phi)$$

1.3 Old encoding

1.3.1 Strategy

- If there is a τ, put the starts of both processes onto it
- Then, append the rest of both processes onto that as well
- Then add any processes without a τ on to the start

1.4 Way to filter out processes with tau

Add new communications

- $a_{\text{first}}|b_{\text{second}} = a_{\text{second}}$
- $\text{skip}|\tau = \tau$
- $\tau_s|a_s = \tau_s + a$

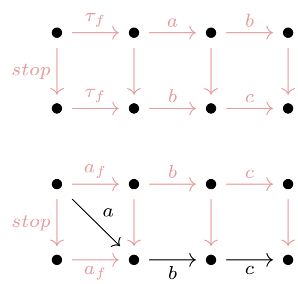
Define function stop with the communications

$$\text{stop}|a = a, \quad a \neq \tau$$

The function

$$\partial_{H_0}(\rho_{f_1}[\partial_{H_1}(p||\text{first}(\text{next}^\infty))]||\text{stop})$$

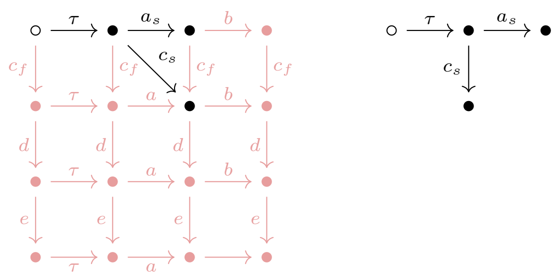
will return an empty process if the first action is τ otherwise return the process



1.5 Cut Tau Tree

Recall: $a_{\text{first}}|b_{\text{second}} = a_{\text{second}}$

$$\rho_{f_1}(\partial_{H_1}(p||\text{first}(\text{second}(\text{next}^\infty)))) \;||\; \rho_{f_1}(\partial_{H_1}(q||\text{first}(\text{second}(\text{next}^\infty))))$$



If neither process has τ then this will just do nothing since the a_{first} will just get deleted at the end
Red parts are just ignoring the irrelevant bits that don't have any further way of connecting

1.6 Reinserting the remaining states of P and Q

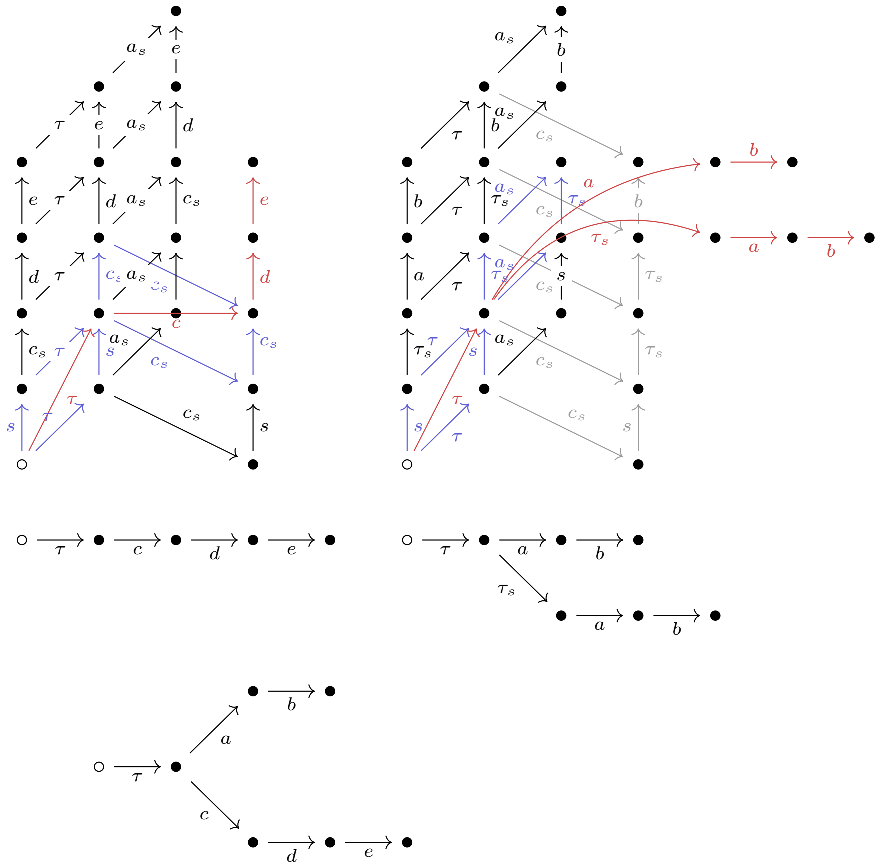
Recall: $\text{skip}|\tau = \tau$, $\tau_s|a_s = \tau_s + a$

- things going up: both process with skip on the start and also prefixed

$$\rho_{f_1}(\partial_{H_1}(\text{skip}(p)||\text{first}(\text{second}(\text{next}^{\infty}))))$$

$$= \text{skip}.a_{\text{second}}.a_2.a_3.\ldots$$

- blue means a successful communication
- red means a communication path



1.7 Final conversion

Define

$$\Gamma_1(p) = \rho_{f_1}(\partial_{H_1}(\text{skip}(p)||\text{first}(\text{next}^{\infty})))$$
$$\Gamma_2(p) = \rho_{f_1}(\partial_{H_1}(\text{skip}(p)||\text{first}(\text{second}(\text{next}^{\infty}))))$$

An encoding of the square operator in ACP is:

$$P\Box Q = \partial_{H_0}[\underbrace{(\Gamma_1(P)\ ||\ \text{stop})}_{\text{Filter from P}} + \underbrace{(\Gamma_2(Q)\ ||\ \text{stop})}_{\text{Filter from Q}} + (\underbrace{(\Gamma_2(P)\ ||\ \Gamma_2(Q))}_{\text{Cut Tau Tree}}\ ||\ \underbrace{\Gamma_2(\text{skip}(P)}_{\text{Reinsert P}}\ ||\ \underbrace{\Gamma_2(\text{skip}(Q))}_{\text{Reinsert Q}}))]$$