UNIVERSITY OF SUSSEX

COMPUTER SCIENCE

A typed, prioritized process algebra

Alan Jeffrey

Computer Science
School of Cognitive and Computing Sciences
University of Sussex
Brighton BN1 9QH

A typed, prioritized process algebra

ALAN JEFFREY

ABSTRACT. This paper describe a typed, prioritized process algebra PPA based on the untyped algebra ACP$_\theta$. We give this an operational semantics, and provide a complete axiomatization for prioritized bisimulation of closed finite terms. Guarded recursion is added to PPA to give μPPA, and we show that every guarded recursion has a unique fixed point. We provide a translation for CLEAVELAND and HENNESSY's prioritized CCS, a subset of BAETEN, BERGSTRA and KLOP's ACP$_\theta$, and a subset of HANSSON and ORAVA's PC into PPA. The only operators which we cannot translate into PPA are those which do not respect prioritized bisimulation.

1 Introduction

Process algebra is a methodology for reasoning about concurrent systems. In MILNER's (1989) *Calculus of Communicating Systems* (CCS) one can describe concurrent systems and their specifications as *agents*, and the task of showing that a system meets its specification becomes that of showing a bisimulation $SPEC \sim SYS$. For example, we could specify a sink process, that will accept arbitrarily many *go* actions until it receives a *stop* signal as:

$$SPEC \stackrel{\text{def}}{=} go.SPEC + stop.\tau.\mathbf{0}$$

We could then attempt to implement this using a sink which can be interrupted, in parallel with an interrupt handler:

$$SYS \stackrel{\text{def}}{=} (SINK \mid HANDLER) \setminus i$$
$$SINK \stackrel{\text{def}}{=} go.SINK + i.\mathbf{0}$$
$$HANDLER \stackrel{\text{def}}{=} stop.\bar{i}.\mathbf{0}$$

Unfortunately, $SPEC \not\sim SYS$ since after a *stop* action the internal interrupt can be delayed indefinitely, for example:

$$SYS \xrightarrow{stop} \xrightarrow{go}$$
$$SPEC \xrightarrow{stop} \xcancel{\xrightarrow{go}}$$

We would like to specify that the interrupt action i has higher priority than the *go* action, and so the only transition SYS can make after a *stop* action is the internal

i. We can do this in CLEAVELAND and HENNESSY's (1988) prioritized variant of CCS where actions are either of low priority (a) or high priority ($\underline{a}$). Then we can respecify *SPEC* and *SYS* with a high priority interrupt:

$$SPEC \stackrel{\text{def}}{=} go.SPEC + stop.\underline{\tau}.\mathbf{0}$$

$$SYS \stackrel{\text{def}}{=} (SINK \mid HANDLER) \setminus i$$

$$SINK \stackrel{\text{def}}{=} go.SINK + \underline{i}.\mathbf{0}$$

$$HANDLER \stackrel{\text{def}}{=} stop.\overline{\underline{i}}.\mathbf{0}$$

Then we can show *SPEC* $\sim$ *SYS*, since the expansion law gives:

$$SYS \sim go.SYS + stop.SYS'$$

$$SYS' \sim go.SYS' + \underline{\tau}.\mathbf{0}$$

where:

$$SYS' \stackrel{\text{def}}{=} (SINK \mid \overline{\underline{i}}.\mathbf{0}) \setminus i$$

Then we can apply CLEAVELAND and HENNESSY's priority law, which says that high priority τ actions have priority over low priority actions:

$$\alpha.t + \underline{\tau}.u \sim \underline{\tau}.u$$

This gives:

$$SYS' \sim \underline{\tau}.\mathbf{0}$$

Thus *SPEC* satisfies the same guarded equations as *SYS* so:

$$SPEC \sim SYS$$

The crucial step in this reasoning is the application of the priority law. Many other prioritized calculi have a similar law. For example, BAETEN, BERGSTRA and KLOP's (1986) ACP$_\theta$ has the law:

$$a.p + b.q \sim_\theta b.q \qquad (a < b)$$

where $\sim_\theta$ is defined in Section 3. HANSSON and ORAVA's (1992) PC has:

$$\tau(\pi).P + \tau(\pi').Q \sim_\pi \tau(\pi').Q \qquad (\pi < \pi')$$

These laws are all examples of a general law:

$$a.P + b.Q \sim_\theta b.Q \qquad (a < b)$$

where $<$ is a strict partial order on actions. For example:

- In CLEAVELAND and HENNESSY's prioritized CCS, $\alpha < \underline{\tau}$.
- In BAETEN, BERGSTRA and KLOP's ACP$_\theta$, $<$ is part of the language.
- In HANSSON and ORAVA's PC, the order $<$ is complex but includes $\tau(\pi) < \tau(\pi')$ if $\pi < \pi'$, discussed further in Appendix C.

HANSSON and ORAVA observed that one may require a partial order rather than a total order for priority, if one is interested in modelling highly distributed systems. For example, in a totally ordered alphabet $\mathbf{A}$ where $a < b$, we can model livelock as:

$$a^{\omega} \,||\, b^{\omega}$$

where:

$$P^{\omega} \stackrel{\text{def}}{=} P.P^{\omega}$$

Then we can prove that:

$$a^{\omega} \,||\, b^{\omega} \sim_{\theta} (a+b)^{\omega} \sim_{\theta} b^{\omega}$$

The a action is never performed, as it is always beaten by the higher priorty b action. This is fine if we wish to model systems where there is only one scheduler, but in a highly distributed system, a^{ω} and b^{ω} may be running on different processors, and so the a action might still be scheduled. In order to model this, we can use the alphabet relabelling constructor $\rho_f(P)$ to inject the processes a^{ω} and b^{ω} from alphabet $\mathbf{A}$ to $\mathbf{A} + \mathbf{A}$:

$$\rho_{\text{inl}}(a^{\omega}) \,||\, \rho_{\text{inr}}(b^{\omega})$$

Then since $\text{inl}\,a \not\leq \text{inr}\,b$ we have:

$$\rho_{\text{inl}}(a^{\omega}) \,||\, \rho_{\text{inr}}(b^{\omega}) \sim_{\theta} (\text{inl}\,a)^{\omega} \,||\, (\text{inr}\,b)^{\omega} \sim_{\theta} (\text{inl}\,a + \text{inr}\,b)^{\omega} \not\sim_{\theta} (\text{inr}\,b)^{\omega}$$

Thus, by using a partial order $\mathbf{A} + \mathbf{A}$, rather than a total order $\mathbf{A}$, we can model a highly distributed system, rather than a one-scheduler system. This is similar to the author's (1992b) use of partial ordered time domains to represent highly distributed timed systems.

In this paper, we:

- Present a typed prioritized process algebra $\text{PPA}(A)$ based on BAETEN, BERGSTRA and KLOP's (1986) ACP_{θ}. The important difference between PPA and ACP_{θ} is that PPA is typed—if A is a strict partial order, then $\text{PPA}(A)$ is a process algebra with alphabet A. For example, if $f : A \to B$ and P is of type $\text{PPA}(A)$ then $\rho_f(P)$ is of type $\text{PPA}(B)$. Although this adds the complexity of a type inference system, it means that prioritized bisimulation is a congruence for PPA.

- Provide $\text{PPA}(A)$ with an operational semantics and a complete axiomatization for prioritized bisimulation of closed finite terms.

- Add guarded recursion into $\text{PPA}(A)$ to get $\mu\text{PPA}(A)$, and show that all guarded recursions have a unique solution, up to prioritized strong bisimulation.

- Show that all the combinators from the above process algebras that respect prioritized bisimulation can be translated into μPPA. Prioritized bisimulation

is a congruence for μPPA, and so the combinators which do not respect it cannot be translated into μPPA.

Although we are interested in the prioritized bisimulation equivalence $P \sim_\theta Q$, it is simpler to define it via the unprioritized bisimulation $P \sim Q$. In many cases, it is possible to show that two processes are bisimilar without taking priority into account, and so it is useful to define $P \sim Q$. An analogy could be made with MILNER's (1989) use of strong and weak bisimulation: weak bisimulation is the equivalence of interest, but many proofs can be performed more easily with strong bisimulation.

This paper is a first step towards finding an expressive language for prioritized process algebra. Further work could involve defining a format for prioritized process algebra similar to GROOTE and VAANDRAGER's (1989) *nyft* or DE SIMONE's (1985) format, and showing that any operator given in such a style could be translated into an appropriate target language.

In this paper, we are only considering algebras where priority is associated with *actions*. There are other algebras, such as CAMILLERI's (1990) CCS with prioritized sum, where priorities are associated with *processes*. It remains to be seen whether there is a framework in which both prioritized actions and prioritized processes can be considered.

2   Strict posets

Our model of prioritized actions is a strict poset $A = (U_A, <_A)$, that is a set U_A together with a transitive irreflexive relation $<_A \subseteq U_A \times U_A$. We shall write $a : A$ for $a \in U_A$. A *morphism* $f : A \to B$ is a function $f \in U_A \to U_B$ such that:

$$a <_A b \Rightarrow fa <_B fb$$

We shall use morphisms to model relabellings: if $f : A \to B$ and P is from PPA(A) then $\rho_f(P)$ is from PPA(B). Two strict posets A and B are *isomorphic* (written $A \simeq B$) iff we can find:

$$f : A \to B \qquad g : B \to A \qquad f \circ g = \mathrm{id} \qquad g \circ f = \mathrm{id}$$

Given strict posets A and B, we can define the *product* and *coproduct* strict posets as the smallest strict posets such that:

where $<_{A \times B}$ and $<_{A+B}$ are the smallest strict posets such that:

$$U_{A \times B} = \{(a,b) \mid a \in U_A, b \in U_B\}$$
$$a <_A a', b <_B b' \Rightarrow (a,b) <_{A \times B} (a',b')$$
$$U_{A+B} = \{\mathrm{inl}\, a \mid a \in U_A\} \cup \{\mathrm{inl}\, b \mid b \in U_B\}$$
$$a <_A a' \Rightarrow \mathrm{inl}\, a <_{A+B} \mathrm{inl}\, a'$$

$$b <_A b' \Rightarrow \mathrm{inr}\, b <_{A+B} \mathrm{inr}\, b'$$

Thus we have morphisms:

$$\mathrm{inl} : A \rightarrow A + B \qquad \mathrm{inr} : B \rightarrow A + B$$
$$\mathrm{fst} : A \times B \rightarrow A \qquad \mathrm{snd} : A \times B \rightarrow B$$

A set $H \subseteq U_A$ is *lower-closed* iff:

$$\forall a \in H . \forall b <_A a. \Rightarrow b \in H$$

We will write $H : \mathrm{Low}(A)$ if H is lower-closed. The strict poset $\partial_H A$ is defined:

$$U_{\partial_H A} = U_A \setminus H$$
$$a <_A b \Leftrightarrow a <_{\partial_H A} b$$

For example, let **2** be the two-point poset with elements $0 <_2 1$. Then $\{0\}$ is lower-closed, and $\partial_{\{0\}}\mathbf{2}$ is the one-point poset with element 1. We shall use ∂_H to model restriction: if P is from $\mathrm{PPA}(A)$ and H is lower-closed then $\partial_H(P)$ is from $\mathrm{PPA}(\partial_H A)$.

$A \mid B$ is the smallest strict poset such that:

$$U_{A\mid B} = (U_A + U_B) \cup (U_A \times U_B)$$
$$a <_A a' \Rightarrow \mathrm{inl}\, a <_{A\mid B} \mathrm{inl}\, a' >_{A\mid B} (a,b)$$
$$b <_B b' \Rightarrow \mathrm{inr}\, b <_{A\mid B} \mathrm{inr}\, b' >_{A\mid B} (a,b)$$

We shall use this to model concurrency: if P is from $\mathrm{PPA}(A)$ and Q is from $\mathrm{PPA}(B)$ then $P \mid Q$ is from $\mathrm{PPA}(A \mid B)$. Note that $A \mid B$ is not commutative or associative, but is commutative and associative *up to isomorphism*, that is:

$$A \mid B \simeq B \mid A \qquad A \mid (B \mid C) \simeq (A \mid B) \mid C$$

This strict poset is similar to WINSKEL's (1984) *synchronization algebra* and is necessary if we are to model algebras such as HANSSON and ORAVA's PC, where concurrency is only commutative and associative up to isomorphism.

3   Prioritized process algebra

We can now define our typed variant of ACP$_\theta$. In order to remain inside classical set theory, we shall assume a universal set of actions U, which is closed under product and coproduct, that is $U \times U \subseteq U$ and $U + U \subseteq U$. Then the untyped language PPA is defined:

$$P ::= a \mid \alpha \mid P.P \mid P + P \mid P \mid P \mid \rho_f P \mid \partial_H P \mid \theta P \mid x$$

where:

- a ranges over U.
- α ranges over $\{\delta, \epsilon\}$.

- f ranges over partial functions from U to U.
- H ranges over subsets of U.
- x ranges over an infinite set of variables $\mathbf{V}$.

We shall write $P = Q$ for syntactic identity between processes.

The process constructors of PPA are:

- a is the process which performs a then terminates.
- δ is the deadlocked process.
- ε is the successfully terminated process.
- $P.Q$ is the sequential composition of P and Q.
- $P + Q$ is the choice between P and Q.
- $P \mid Q$ is the concurrent composition of P and Q. If P can perform a and Q can perform b then $P \mid Q$ can perform the interleaving $\mathrm{inl}\,a$ or $\mathrm{inr}\,b$, or the synchronization (a, b). This is similar to WINSKEL's (1984) synchronization algebra, except that we do not have a 0 action to represent deadlock.
- $\rho_f P$ applies f to the actions of P, so $\rho_{\mathrm{succ}}(1.2.3)$ is 2.3.4.
- $\partial_H P$ stops P performing actions in H, so $\partial_{\{b\}}(a.b.c)$ is $a.\delta$.
- θP prunes P, that is it removes any actions that can be beaten by a higher priority action. For example, if $a <_A b$ then $\theta(a + b)$ is b. This operators semantics depends on the strict poset A which is given by the type information below.
- x is a free variable, used to define recursion in Section 5

PPA is an untyped language, which presents problems for prioritized bisimulation, since the constructors of PPA do not have to respect the priority ordering. For example, if f is not a morphism then we can find $a <_A b$ such that $fa \not<_A fb$, and so:

$$a + b \sim_\theta b \quad \text{but} \quad \rho_f(a + b) \sim \rho_f a + \rho_f b \not\sim_\theta \rho_f b$$

Similarly, if H is not lower-closed then we can find $a <_A b$ such that $a \notin H$ and $b \in H$, and so:

$$a + b \sim_\theta b \quad \text{but} \quad \partial_H(a + b) \sim \partial_H a + \partial_H b \sim a \not\sim_\theta \partial_H(b)$$

Our type judgements for PPA are given in Table 1 as judgements of the form:

$$\Gamma \vdash P : \mathrm{PPA}(A)$$

where:

- Γ is a *context* of the form $x_1 : \mathrm{PPA}(A_1), \ldots, x_n : \mathrm{PPA}(A_n)$ and each of the x_i are distinct variables.
- P is a process.
- A is a strict poset with $U_A \subseteq U$.

$$\frac{a : A}{\vdash a : \text{PPA}(A)} \quad \overline{\vdash \delta : \text{PPA}(A)} \quad \overline{\vdash \varepsilon : \text{PPA}(A)}$$

$$\frac{\Gamma \vdash P : \text{PPA}(A) \quad \Gamma \vdash Q : \text{PPA}(A)}{\Gamma \vdash P.Q : \text{PPA}(A)}$$

$$\frac{\Gamma \vdash P : \text{PPA}(A) \quad \Gamma \vdash Q : \text{PPA}(A)}{\Gamma \vdash P + Q : \text{PPA}(A)}$$

$$\frac{\Gamma \vdash P : \text{PPA}(A) \quad \Gamma \vdash Q : \text{PPA}(B)}{\Gamma \vdash P \mid Q : \text{PPA}(A \mid B)}$$

$$\frac{f : A \to B \quad \Gamma \vdash P : \text{PPA}(A)}{\Gamma \vdash \rho_f P : \text{PPA}(B)} \quad \frac{H : \text{Low}(A) \quad \Gamma \vdash P : \text{PPA}(A)}{\Gamma \vdash \partial_H P : \text{PPA}(\partial_H A)}$$

$$\frac{\Gamma \vdash P : \text{PPA}(A)}{\Gamma \vdash \theta P : \text{PPA}(A)}$$

$$\overline{x : \text{PPA}(A) \vdash x : \text{PPA}(A)}$$

$$\frac{\Gamma, x : \text{PPA}(A), y : \text{PPA}(B), \Delta \vdash P : \text{PPA}(A)}{\Gamma, y : \text{PPA}(B), x : \text{PPA}(A), \Delta \vdash P : \text{PPA}(A)}$$

$$\frac{\Gamma \vdash P : \text{PPA}(A)}{\Gamma, x : \text{PPA}(B) \vdash P : \text{PPA}(A)} \; [x \text{ not in } \Gamma]$$

TABLE 1. The type rules for PPA

The context Γ is used to give the type of any free variables in P, for example:

$$x : \text{PPA}(A), y : \text{PPA}(B) \vdash x \mid y : \text{PPA}(A \mid B)$$

Note that the relabelling constructor $\rho_f P$ is allowed to change the priorities of P. For example, if $\vdash P : \text{PPA}(A)$ is unprioritized (that is $<_A = \emptyset$) and $f : A \to \mathbf{N}$ assigns a priority to each action then:

$$\vec{f} \; : \; A \to A \times \mathbf{N}$$
$$\vec{f} a = (a, fa)$$

is a morphism, and so $\rho_{\vec{f}} P$ prioritizes P. Thus, if we want to use an unprioritized component in a prioritized system, we can verify it in an unprioritized setting, and then embed it using a morphism. This is similar to SCHNEIDER's (1990) approach to embedding untimed processes into timed settings using timewise refi nements.

As another example, we could model a UNIX process as a process from $\text{PPA}(A \times \mathbf{N})$ where (a, n) has priority n. Then the `nice(1)` n operator which decreases the priority of a process by n could be defi ned:

$$\rho_{\text{subtract}_n} P$$

where:

$$\text{subtract}_n(a, m) = (a, m - n)$$

Alan Jeffrey

$$\frac{}{\varepsilon\surd} \qquad \frac{P\surd \quad Q\surd}{(P.Q)\surd} \qquad \frac{P\surd}{(P+Q)\surd} \qquad \frac{Q\surd}{(P+Q)\surd} \qquad \frac{P\surd \quad Q\surd}{(P\,|\,Q)\surd} \qquad \frac{P\surd}{(\rho_f P)\surd} \qquad \frac{P\surd}{(\partial_H P)\surd} \qquad \frac{P\surd}{(\theta P)\surd}$$

TABLE 2. The termination of $\text{PPA}(A)$

$$\frac{}{a \xrightarrow{a} \varepsilon} \qquad \frac{P \xrightarrow{a} P'}{P.Q \xrightarrow{a} P'.Q} \qquad \frac{P\surd \quad Q \xrightarrow{a} Q'}{P.Q \xrightarrow{a} Q'}$$

$$\frac{P \xrightarrow{a} P'}{P+Q \xrightarrow{a} P'} \qquad \frac{Q \xrightarrow{b} Q'}{P+Q \xrightarrow{b} Q'}$$

$$\frac{P \xrightarrow{a} P'}{P\,|\,Q \xrightarrow{\text{in}la} P'\,|\,Q} \qquad \frac{Q \xrightarrow{b} Q'}{P\,|\,Q \xrightarrow{\text{in}rb} P\,|\,Q'} \qquad \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{b} Q'}{P\,|\,Q \xrightarrow{(a,b)} P'\,|\,Q'}$$

$$\frac{P \xrightarrow{a} P'}{\rho_f P \xrightarrow{fa} \rho_f P'} \qquad \frac{P \xrightarrow{a} P'}{\partial_H P \xrightarrow{a} \partial_H P'} \; [a \notin H]$$

$$\frac{P \xrightarrow{a} P' \quad P \downarrow a}{\theta P \xrightarrow{a} \theta P'}$$

TABLE 3. The operational semantics of $\text{PPA}(A)$

The operational semantics of $\text{PPA}(A)$ is similar to that of ACP, and is given in Tables 2 and 3. In particular, the semantics of sequential composition $P.Q$ uses a predicate $P\surd$, which is true iff P can terminate—this is similar to BAETEN and WEIJLAND's (1990, Section 3.3.5) $\surd P$ from PA_ε. The semantics for θP allows $\theta P \xrightarrow{a} \theta P'$ if $P \downarrow a$, where:

$$P \downarrow a \quad \text{iff} \quad \forall b > a . P \not\xrightarrow{b}$$

Note that the priority order is only used in the pruning operator θP. For example, $a.P + b.Q \xrightarrow{a} P$ even if $a < b$, but $\theta(a.P + b.Q) \not\xrightarrow{a}$. Note also that since there is no recursion in PPA we can show by induction on P that the relation $P \xrightarrow{a} P'$ is well-defined.

We can then define MILNER's (1989) *strong bisimulation*, modified to take typing and termination into account. $\mathcal{R}$ is a strong $\text{PPA}(A)$-bisimulation if, whenever $P \mathrel{\mathcal{R}} Q$:

- $\vdash P : \text{PPA}(A)$ and $\vdash Q : \text{PPA}(A)$,
- If $P \xrightarrow{a} P'$ then $Q \xrightarrow{a} Q'$ and $P' \mathrel{\mathcal{R}} Q'$.
- If $Q \xrightarrow{a} Q'$ then $P \xrightarrow{a} P'$ and $P' \mathrel{\mathcal{R}} Q'$.
- $P\surd$ iff $Q\surd$.

We shall write $\vdash P \sim Q : \text{PPA}(A)$ iff there is a strong $\text{PPA}(A)$-bisimulation $\mathcal{R}$ such that $P \mathrel{\mathcal{R}} Q$. For example, the identity relation I is a strong PPA-bisimulation, and so $\vdash P \sim P : \text{PPA}(A)$. As another example, if $a < b$ then the only transition of $\theta(a.P + b.Q)$ is $\theta(a.P + b.Q) \xrightarrow{a} \theta P$, and so it is simple to show that the relation:

$$\{(\theta(a.P + b.Q), \theta(a.P))\} \cup I$$

is a strong bisimulation, and so $\theta(a.P + b.Q) \sim \theta(a.P)$. Thus, $\vdash a.P + b.Q \sim_\theta a.P : \text{PPA}(A)$ if we define:

$$\vdash P \sim_\theta Q : \text{PPA}(A) \quad \text{iff} \quad \vdash \theta P \sim \theta Q : \text{PPA}(A)$$

This paper is more concerned with prioritized bisimulation $\sim_\theta$ than with unprioritized bisimulation $\sim$. However, it is often simpler to prove properites about $\sim$ than about $\sim_\theta$. The rest of this section will be taken up by proving that $\sim_\theta$ is a congruence.

PROPOSITION 1. *If $\Gamma \vdash P : \text{PPA}(A)$ and x is free in P then x is in Γ.*

PROOF. An induction on the proof of $\Gamma \vdash P : \text{PPA}(A)$. $\qquad\qquad\square$

Let a *substitution* σ be a function $\sigma : \mathbf{V} \to \text{PPA}$ which is almost everywhere the identity. Let $(x_1 := P_1, \ldots, x_n := P_n)$ be the substitution which maps x_i to P_i and anything else to itself. Define $P[\sigma]$ as the standard replacement of x by σx with appropriate α-conversion of bound variables. We can extend typing, bisimulation and prioritized bisimulation to open terms and substitutions in the obvious pointwise fashion:

$$\Gamma \vdash \sigma : \Delta \quad \text{iff} \quad \forall i \,.\, \Gamma \vdash P_i : \text{PPA}(A_i)$$
$$\Gamma \vdash P \sim Q : \text{PPA}(A) \quad \text{iff} \quad \forall \vdash \sigma : \Gamma \,.\, \vdash P[\sigma] \sim Q[\sigma] : \text{PPA}(A)$$
$$\Gamma \vdash P \sim_\theta Q : \text{PPA}(A) \quad \text{iff} \quad \Gamma \vdash \theta P \sim \theta Q : \text{PPA}(A)$$
$$\Gamma \vdash \sigma \sim \sigma' : \Delta \quad \text{iff} \quad \forall i \,.\, \Gamma \vdash P_i \sim P_i' : \text{PPA}(A_i)$$
$$\Gamma \vdash \sigma \sim_\theta \sigma' : \Delta \quad \text{iff} \quad \Gamma \vdash \theta\sigma \sim \theta\sigma' : \Delta$$

where:

$$\Delta = (x_1 : \text{PPA}(A_1), \ldots, x_n : \text{PPA}(A_n))$$
$$\sigma = (x_1 := P_1, \ldots, x_n := P_n)$$
$$\sigma' = (x_1 := P_1', \ldots, x_n := P_n')$$
$$\theta\sigma = (x_1 := \theta P_1, \ldots, x_n := \theta P_n)$$

Then we can show that our type system respects substitution.

PROPOSITION 2. *If $\Gamma \vdash P : \text{PPA}(A)$ and $\Delta \vdash \sigma : \Gamma$ then $\Delta \vdash P[\sigma] : \text{PPA}(A)$.*

PROOF. An induction on the proof of $\Gamma \vdash P : \text{PPA}(A)$. $\qquad\qquad\square$

Note that this means the type system supports contraction:

$$\frac{\Gamma, x : \text{PPA}(A), y : \text{PPA}(A) \vdash P : \text{PPA}(B)}{\Gamma, z : \text{PPA}(A) \vdash P[x := z, y := z] : \text{PPA}(B)}$$

and cut:

$$\frac{\Gamma \vdash P : \text{PPA}(A) \quad \Delta, x : \text{PPA}(A) \vdash Q : \text{PPA}(B)}{\Gamma, \Delta \vdash Q[x := P] : \text{PPA}(B)}$$

PROPOSITION 3. $\sim$ *is a congruence.*

PROOF. Define:

$$\mathcal{R} = \{(P[\sigma], P[\sigma']) \mid \Gamma \vdash P : \text{PPA}(A), \vdash \sigma \sim \sigma' : \Gamma\}$$

Then we can prove by induction on P that $\mathcal{R}$ is a $\text{PPA}(A)$-strong bisimulation. Thus, if $\Gamma \vdash P : \text{PPA}(A)$ and $\vdash \sigma \sim \sigma' : \Gamma$ then $\vdash P[\sigma] \sim P[\sigma'] : \text{PPA}(A)$. If $\Gamma \vdash P : \text{PPA}(A)$ then:

$$\begin{aligned}
\Delta \vdash \sigma \sim \sigma' : \Gamma &\Rightarrow \forall \vdash \rho : \Delta . \sigma[\rho] \sim \sigma'[\rho] : \Gamma \\
&\Rightarrow \forall \vdash \rho : \Delta . P[\sigma[\rho]] \sim P[\sigma'[\rho]] : \text{PPA}(A) \\
&\Rightarrow \forall \vdash \rho : \Delta . P[\sigma][\rho] \sim P[\sigma'][\rho] : \text{PPA}(A) \\
&\Rightarrow \Delta \vdash P[\sigma] \sim P[\sigma'] : \text{PPA}(A)
\end{aligned}$$

And so $\sim$ is a congruence.   □

PROPOSITION 4.   *If* $\Gamma \vdash P : \text{PPA}(A)$ *and* $\Delta \vdash \sigma : \Gamma$ *then* $\Delta \vdash \theta P[\sigma] \sim \theta P[\theta\sigma] :$ $\text{PPA}(A)$.

PROOF. Since $\text{PPA}(A)$ is image-finite, we can show that $\theta P \downarrow a$ iff $P \downarrow a$. Using this, we can show the following, by exhibiting appropriate bisimulations:

$$\begin{aligned}
\Gamma &\vdash \theta(P + Q) \sim \theta(\theta P + \theta Q) : \text{PPA}(A) \\
\Gamma &\vdash \theta(P.Q) \sim \theta(\theta P.\theta Q) : \text{PPA}(A) \\
\Gamma &\vdash \theta(P \mid Q) \sim \theta(\theta P \mid \theta Q) : \text{PPA}(A \mid B) \\
\Gamma &\vdash \theta \rho_f P \sim \theta \rho_f \theta P : \text{PPA}(A) \\
\Gamma &\vdash \theta \partial_H P \sim \theta \partial_H \theta P : \text{PPA}(A) \\
\Gamma &\vdash \theta P \sim \theta \theta P : \text{PPA}(A)
\end{aligned}$$

From this, the result follows by structural induction on P.   □

PROPOSITION 5. $\sim_\theta$ *is a congruence.*

PROOF. For any $\Gamma \vdash P : \text{PPA}(A)$:

$$\begin{aligned}
\Delta \vdash \sigma \sim_\theta \sigma' : \Gamma &\Rightarrow \Delta \vdash \theta\sigma \sim \theta\sigma' : \Gamma \\
&\Rightarrow \Delta \vdash \theta P[\theta\sigma] \sim \theta P[\theta\sigma'] : \text{PPA}(A) \\
&\Rightarrow \Delta \vdash \theta P[\sigma] \sim \theta P[\sigma'] : \text{PPA}(A) \\
&\Rightarrow \Delta \vdash P[\sigma] \sim_\theta P[\sigma'] : \text{PPA}(A)
\end{aligned}$$

Thus $\sim_\theta$ is a congruence.   □

Note that the proofs of Propositions 4 and 5 depend on the use of strong bisimulation and the pruning operation θP. If we were to define prioritized bisimulation directly without the use of θP, then these proofs would be much harder.

$$\theta P = P \qquad a.P + b.Q = b.Q \quad (a < b) \qquad P + \delta = P$$
$$\epsilon.P = P \qquad (P + Q).R = P.R + Q.R \qquad P + P = P$$
$$\delta.P = \delta \qquad P + (Q + R) = (P + Q) + R \qquad P + Q = Q + P$$
$$\rho_f a = fa \qquad \rho_f(P.Q) = (\rho_f P).(\rho_f Q) \qquad (P.Q).R = P.(Q.R)$$
$$\rho_f \alpha = \alpha \qquad \rho_f(P + Q) = (\rho_f P) + (\rho_f Q) \qquad \partial_H a = \delta \quad (a \in H)$$
$$\partial_H \alpha = \alpha \qquad \partial_H(P.Q) = (\partial_H P).(\partial_H Q) \qquad \partial_H a = a \quad (a \notin H)$$
$$\alpha | \delta = \delta \qquad \partial_H(P + Q) = (\partial_H P) + (\partial_H Q) \qquad \alpha | \epsilon = \alpha$$

$$P = \sum\{a_i.P_i \mid i \in I\} + \alpha$$
$$Q = \sum\{b_j.Q_j \mid j \in J\} + \beta$$
$$\Rightarrow P \mid Q = \sum\{(\mathrm{inl}\, a_i).(P_i \mid Q) \mid i \in I\}$$
$$+ \sum\{(\mathrm{inr}\, b_j).(P \mid Q_j) \mid j \in J\}$$
$$+ \sum\{(a_i, b_j).(P_i \mid Q_j) \mid i \in I, j \in J\}$$
$$+ \alpha | \beta$$

TABLE 4. The axiomatization of $\mathrm{PPA}(A)$

4   Complete axiomatization of PPA

Let us write $\Gamma \vdash P = Q : \mathrm{PPA}(A)$ iff $\Gamma \vdash P : \mathrm{PPA}(A)$, $\Gamma \vdash Q : \mathrm{PPA}(A)$ and $P = Q$ can be proved from the axioms in Table 4 together with standard equational reasoning. In this Section, we shall show that these axioms are sound and complete for prioritized bisimulation.

PROPOSITION 6.   *If* $\vdash P = Q : \mathrm{PPA}(A)$ *then* $\vdash P \sim_\theta Q : \mathrm{PPA}(A)$.

PROOF.   Provide a bisimulation for each axiom.   □

Let $\vdash P : \mathrm{PPA}(A)$ be $\mathrm{PPA}(A)$-pre-normal iff:

$$P = \sum\{a_i.P_i \mid i \in I\} + \alpha$$

where all the P_i are $\mathrm{PPA}(A)$-pre-normal and α ranges over $\{\delta, \epsilon\}$. P is $\mathrm{PPA}(A)$-normal iff all the a_i are $<_A$-incomparable and all the P_i are $\mathrm{PPA}(A)$-normal. $\vdash P : \mathrm{PPA}(A)$ can be (pre-)normalized iff there is a $\mathrm{PPA}(A)$-(pre-)normal P' such that $\vdash P = P' : \mathrm{PPA}(A)$.

PROPOSITION 7.   *If P and Q are $\mathrm{PPA}(A)$-pre-normal then the following can be $\mathrm{PPA}(A)$-pre-normalized: $P.Q$, $P + Q$, $P \mid Q$, $\rho_f P$, $\partial_H P$, θP.*

PROOF (BY INDUCTION ON P AND Q).   We shall prove the case for $P \mid Q$, and the others are similar. Let:

$$P = \sum\{a_i.P_i \mid i \in I\} + \alpha$$
$$Q = \sum\{b_j.Q_j \mid j \in J\} + \beta$$

Then let $\gamma = \varepsilon$ if $\alpha = \beta = \varepsilon$ and let $\gamma = \delta$ otherwise. Then:

$$\vdash P \mid Q = \Sigma\{(\mathrm{inl}\,a_i).(P_i \mid Q) \mid i \in I\}$$
$$+ \Sigma\{(\mathrm{inr}\,b_j).(P \mid Q_j) \mid j \in J\}$$
$$+ \Sigma\{(a_i, b_j).(P_i \mid Q_j) \mid i \in I, j \in J\}$$
$$+ \gamma : \mathrm{PPA}(A)$$

By induction, $P_i \mid Q$, $P \mid Q_j$ and $P_i \mid Q_j$ can be $\mathrm{PPA}(A)$-pre-normalized, and so $P \mid Q$ can be. $\qquad\square$

PROPOSITION 8. *Any* $\vdash P : \mathrm{PPA}(A)$, *can be* $\mathrm{PPA}(A)$-*normalized.*

PROOF. An induction on P, using Proposition 7, the laws for $+$ and the priority law $a.P + b.Q = b.Q \quad (a < b)$. $\qquad\square$

PROPOSITION 9. *If* $\vdash P \sim_\theta Q : \mathrm{PPA}(A)$ *and* P *and* Q *are normal, then* $P = Q$.

PROOF (BY STRUCTURAL INDUCTION ON P AND Q). Let $P = \Sigma\mathcal{P} + \alpha$ and $Q = \Sigma\mathcal{Q} + \beta$ where:

$$\mathcal{P} = \{a_i.P_i \mid i \in I\}$$
$$\mathcal{Q} = \{b_j.Q_j \mid j \in J\}$$

Then for any $i \in I$, $\theta P \xrightarrow{a_i} \theta P_i$, so there must be a j such that $\theta Q \xrightarrow{b_j} \theta Q_j$, $a_i = b_j$ and $\theta P_i \sim \theta Q_j$. By induction, $P_i = Q_j$, so $a_i.P_i \in \mathcal{Q}$. Thus, $\mathcal{P} \subseteq \mathcal{Q}$, and by similar reasoning, $\mathcal{Q} \subseteq \mathcal{P}$, so $\mathcal{P} = \mathcal{Q}$. If $\alpha = \delta$ then $\neg P\surd$ so $\neg Q\surd$ so $\alpha = \delta$. If $\alpha = \varepsilon$ then $P\surd$ so $Q\surd$ so $\alpha = \varepsilon$. Thus $\alpha = \beta$ and so $P = Q$. $\qquad\square$

COROLLARY 10. $\vdash P \sim_\theta Q : \mathrm{PPA}(A)$ *iff* $\vdash P = Q : \mathrm{PPA}(A)$.

5   Recursion

So far, we have only dealt with finite processes, and have not considered recursive processes such as *SINK* from Section 1. To allow recursion, we shall add a recursion combinator to $\mathrm{PPA}(A)$ to get $\mu\mathrm{PPA}(A)$. However, we shall only allow *guarded* recursions, since there are some unguarded equations which have no solutions. For example, there can be no process from $\mu\mathrm{PPA}(\mathbf{N})$ which satisfies the equation:

$$P \sim \theta(0 + \rho_{\mathrm{succ}}P)$$

since:

- If $\forall n . P \xrightarrow{n}\!\!\!\!\!/\;$ then $P \downarrow 0$ so $\theta(0 + \rho_{\mathrm{succ}}P) \xrightarrow{0}$ so $P \xrightarrow{0}$.
- If $P \xrightarrow{n}$ then $0 + \rho_{\mathrm{succ}}P \xrightarrow{n+1}$ so $0 + \rho_{\mathrm{succ}}P \not\downarrow n$ so $\theta(0 + \rho_{\mathrm{succ}}P) \xrightarrow{n}\!\!\!\!\!/\;$ so $P \xrightarrow{n}\!\!\!\!\!/\;$.

For this reason, we shall only define guarded recursions in $\mu\mathrm{PPA}(A)$:

$$P ::= \cdots \mid \mu x . P$$

where x is guarded in P, that is any occurrence of x in P is inside a subterm $Q.R$ where x is guarded in Q and $\neg Q\surd$. For example, x is not guarded in $0 + \rho_{\text{succ}}x$. We can extend the type system from $\text{PPA}(A)$ to $\mu\text{PPA}(A)$:

$$\frac{\Gamma, x : \mu\text{PPA}(A) \vdash P : \mu\text{PPA}(A)}{\Gamma \vdash \mu x. P : \mu\text{PPA}(A)}$$

There are two possible ways of extending the operational semantics. We can either allow $\surd$ and $\longrightarrow$ to be defined for open terms, and use:

$$\frac{P\surd}{\mu x. P\surd} \quad \frac{P \xrightarrow{a} P'}{\mu x. P \xrightarrow{a} P'[x := \mu x. P]}$$

Alternatively, we could use MILNER's proof rule for recursion, which does not require us to give a semantics to open terms:

$$\frac{P[x := \mu x. P]\surd'}{\mu x. P\surd'} \quad \frac{P[x := \mu x. P] \xrightarrow{a}' P'}{\mu x. P \xrightarrow{a}' P'}$$

These rules are more standard, but are not obviously well-defined, since the semantics of the recursive term $\mu x. P$ may involve negative premises (in G-ROOTE's (GROOTE, 1989) terms, this provides a *stratification* of the operational semantics). For example, if we were to allow unguarded recursions such as $\mu x. (\theta(0 + \rho_{\text{succ}}x))$ then there would be no transition system $\longrightarrow'$. Fortunately, the above transition systems are equivalent, since we are only dealing with guarded terms.

PROPOSITION 11. *If x is guarded in P then:*

1. $P[x := Q]\surd'$ iff $P\surd'$.
2. $P[x := Q] \xrightarrow{a}' R$ iff $P \xrightarrow{a}' P'$ and $R = P'[x := Q]$.

PROOF.

1. An induction on the proof of $\surd$.
2. An induction on the proof of $\longrightarrow$. $\qquad\square$

COROLLARY 12. $P\surd$ iff $P\surd'$ and $P \xrightarrow{a} P'$ iff $P \xrightarrow{a}' P'$.

This is useful, since $\longrightarrow'$ is a more standard definition, but $\longrightarrow$ is obviously well-defined, since it is given by structural induction on processes. We can define bisimulation and prioritized bisimulation as before, and adapt MILNER's (1989, Proposition 4.12) proof that strong bisimulation is a congruence for $\mu\text{PPA}(A)$.

PROPOSITION 13. $\sim$ *is a congruence for $\mu\text{PPA}(A)$.*

PROOF. Show by induction on P that if $\Gamma \vdash P : \text{PPA}(A)$ and $\Delta \vdash \sigma \sim \sigma' : \Gamma$ then $\Delta \vdash P[\sigma] \sim P[\sigma']$. The only tricky case is if $P = \mu x. Q$. For any $\vdash \rho : \Delta$ and fresh

y:

$$P[\sigma][\rho] = \mu y.\,(Q[x := y][\sigma][\rho]) = \mu y.\,R$$
$$P[\sigma'][\rho] = \mu y.\,(Q[x := y][\sigma'][\rho]) = \mu y.\,R'$$

By induction, $y : \mu\mathrm{PPA}(A) \vdash R \sim R' : \mu\mathrm{PPA}(A)$. Then show by induction on the proof of $\longrightarrow$ and $\sqrt{}$ that for any $y : \mu\mathrm{PPA} \vdash S : \mu\mathrm{PPA}(B)$ that:

- If $S[y := \mu y.\,R] \xrightarrow{a} T$ then $T \sim S'[y := \mu y.\,R]$ and $S[y := \mu y.\,R'] \xrightarrow{a}\sim S'[y := \mu y.\,R']$.
- If $S[y := \mu y.\,R'] \xrightarrow{a} T$ then $T \sim S'[y := \mu y.\,R']$ and $S[y := \mu y.\,R] \xrightarrow{a}\sim S'[y := \mu y.\,R]$.
- $S[y := \mu y.\,R]\sqrt{}$ iff $S[y := \mu y.\,R']\sqrt{}$.

Thus $\mathcal{R}_B$ is a $\mu\mathrm{PPA}(B)$-strong bisimulation, where:

$$\mathcal{R}_B = \{(T,T') \mid y : \mu\mathrm{PPA}(A) \vdash S : \mu\mathrm{PPA}(B)$$
$$\vdash T \sim S[y := \mu y.\,R] : \mu\mathrm{PPA}(B)$$
$$\vdash T' \sim S[y := \mu y.\,R'] : \mu\mathrm{PPA}(B)\}$$

Since $(\mu y.\,R, \mu y.\,R') \in \mathcal{R}_A$ this means $\vdash \mu y.\,R \sim \mu y.\,R' : \mu\mathrm{PPA}(A)$ and so for any $\vdash \rho : \Delta, \vdash P[\sigma][\rho] \sim P[\sigma'][\rho] : \mu\mathrm{PPA}(A)$ and so $\Delta \vdash P[\sigma] \sim P[\sigma'] : \mu\mathrm{PPA}(A)$. $\square$

We can also show that $\mu x.\,P$ is the unique solution to $Q \sim P[x := Q]$.

PROPOSITION 14. *If x is guarded in P and $\Gamma, x : \mu\mathrm{PPA}(A) \vdash P : \mu\mathrm{PPA}(A)$ then:*

1. $\Gamma \vdash \mu x.\,P \sim P[x := \mu x.\,P] : \mu\mathrm{PPA}(A)$.
2. *If $\Gamma \vdash Q \sim P[x := Q] : \mu\mathrm{PPA}(A)$ then $\Gamma \vdash Q \sim \mu x.\,P : \mu\mathrm{PPA}(A)$.*

PROOF.

1. Let $\mathcal{R} = I \cup \{(\mu x.\,P, P[x := \mu x.\,P])\}$ which is a $\mu\mathrm{PPA}(A)$-strong bisimulation, so $\vdash \mu x.\,P \sim P[x := \mu x.\,P] : \mu\mathrm{PPA}(A)$. Then if y is a fresh variable then:

$$\forall \vdash \sigma : \Gamma.\,\mu y.\,(P[x := y][\sigma])$$
$$\sim P[x := y][\sigma][y := \mu y.\,(P[x := y][\sigma])] : \mu\mathrm{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma.\,(\mu x.\,P)[\sigma] \sim P[x := y][\sigma][y := (\mu x.\,P)[\sigma]] : \mu\mathrm{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma.\,(\mu x.\,P)[\sigma] \sim P[x := y][y := (\mu x.\,P)][\sigma] : \mu\mathrm{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma.\,(\mu x.\,P)[\sigma] \sim P[x := (\mu x.\,P)][\sigma] : \mu\mathrm{PPA}(A)$$
$$\Rightarrow \Gamma \vdash (\mu x.\,P) \sim P[x := (\mu x.\,P)] : \mu\mathrm{PPA}(A)$$

   Thus $\mu x.\,P$ is a solution to $Q \sim P[x := Q]$.

2. We shall first show the case when the only free variable of P is x. If $\vdash Q \sim$

$P[x := Q]$ then:

$$\mathcal{R} = \{(P', Q') \mid x : \mu\text{PPA}(A) \vdash R : \mu\text{PPA}(A)$$
$$x \text{ is guarded in } R$$
$$\vdash P' \sim R[x := \mu x . P] : \mu\text{PPA}(A)$$
$$\vdash Q' \sim R[x := Q] : \mu\text{PPA}(A)\}$$

Then for any $P' \mathcal{R} Q'$:

- If $P' \xrightarrow{a} P''$ then $R[x := \mu x . P] \xrightarrow{a} \sim P''$ so by Proposition 11 $R \xrightarrow{a} R'$ and $P'' \sim R'[x := \mu x . P] \sim R'[x := P][x := \mu x . P]$. Since $Q' \sim R[x := Q]$, $Q' \xrightarrow{a} Q'' \sim R'[x := Q] \sim R'[x := P][x := Q]$. Since x is guarded in $R'[x := P]$, $P'' \mathcal{R} Q''$.
- Similarly, if $Q' \xrightarrow{a} Q''$ then $P' \xrightarrow{a} P''$ and $P'' \mathcal{R} Q''$.
- $P'\surd$ iff $R[x := \mu x . P]\surd$ iff $R\surd$ iff $R[x := Q]\surd$ iff $Q'\surd$.

Then $\mathcal{R}$ is a $\mu\text{PPA}(A)$-strong bisimulation, so:

$$\vdash \mu x . P \sim P[x := \mu x . P] \sim P[x := Q] \sim Q : \mu\text{PPA}(A)$$

We can now show the case when P has many free variables. If y is a fresh variable then:

$$\Gamma \vdash Q \sim P[x := Q] : \text{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma . Q[\sigma] \sim P[x := Q][\sigma] : \text{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma . Q[\sigma] \sim P[x := y][y := Q][\sigma] : \text{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma . Q[\sigma] \sim P[x := y][\sigma][y := Q[\sigma]] : \text{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma . Q[\sigma] \sim \mu y . (P[x := y][\sigma]) : \text{PPA}(A)$$
$$\Rightarrow \forall \vdash \sigma : \Gamma . Q[\sigma] \sim (\mu x . P)[\sigma]) : \text{PPA}(A)$$

Thus $\mu x . P$ is the unique solution to $Q \sim P[x := Q]$. $\qquad\qquad \square$

We can now prove the same results about $\sim_\theta$.

PROPOSITION 15. *If* $\Gamma \vdash P : \mu\text{PPA}(A)$ *and* $\Delta \vdash \sigma : \Gamma$ *then* $\Delta \vdash \theta P[\sigma] \sim \theta P[\theta\sigma] : \mu\text{PPA}(A)$.

PROOF (BY INDUCTION ON THE SIZE OF P). The only case different from Proposition 4 is if $P = \mu x . Q$. Then if y is fresh, then:

$$\Delta \vdash \theta(\mu x . Q)[\sigma] \sim \theta Q[x := \mu x . Q][\sigma]$$
$$\sim \theta Q[x := \theta\mu x . Q][\sigma]$$
$$\sim \theta Q[x := y][y := \theta\mu x . Q][\sigma]$$
$$\sim \theta Q[x := y][\sigma][y := \theta(\mu x . Q)[\sigma]] : \mu\text{PPA}(A)$$

and:

$$\Delta \vdash \theta(\mu x . Q)[\theta\sigma] \sim \theta Q[x := \mu x . Q][\theta\sigma]$$
$$\sim \theta Q[x := y][y := \theta\mu x . Q][\theta\sigma]$$
$$\sim \theta Q[x := y][\theta\sigma][y := \theta(\mu x . Q)[\theta\sigma]]$$
$$\sim \theta Q[x := y][\sigma][y := \theta(\mu x . Q)[\theta\sigma]] : \mu\text{PPA}(A)$$

Then by Proposition 14.2:

$$\Delta \vdash \theta(\mu x . Q)[\sigma] \sim \mu y . (\theta Q[x := y][\sigma]) \sim \theta(\mu x . Q)[\theta\sigma] : \mu\text{PPA}(A)$$

as required.   □

This is enough to show that $\mu x . P$ is the unique solution to $Q \sim_\theta P[x := Q]$.

PROPOSITION 16. *If x is guarded in P and $\Gamma, x : \mu\text{PPA}(A) \vdash P : \mu\text{PPA}(A)$ then:*

1. $\Gamma \vdash \mu x . P \sim_\theta P[x := \mu x . P] : \mu\text{PPA}(A)$.
2. *If $\Gamma \vdash Q \sim_\theta P[x := Q] : \mu\text{PPA}(A)$ then $\Gamma \vdash Q \sim_\theta \mu x . P : \mu\text{PPA}(A)$.*

PROOF.

1. By Proposition 14.1, $\Gamma \vdash \theta\mu x . P \sim \theta P[x := \mu x . P] : \mu\text{PPA}(A)$ so $\Gamma \vdash \mu x . P \sim_\theta P[x := \mu x . P]$.

2. By Proposition 15:

$$\Gamma \vdash \theta Q \sim \theta P[x := Q] \sim \theta P[x := \theta Q] : \mu\text{PPA}(A)$$

   Similarly:

$$\Gamma \vdash \theta\mu x . P \sim \theta P[x := \theta\mu x . P] : \mu\text{PPA}(A)$$

   So by Proposition 14.2:

$$\Gamma \vdash \theta Q \sim \mu x . \theta P \sim \theta\mu x . P : \mu\text{PPA}(A)$$

   So $\Gamma \vdash Q \sim_\theta \mu x . P : \mu\text{PPA}(A)$.   □

Finally, we can show that $\sim_\theta$ is a congruence for $\mu\text{PPA}(A)$.

PROPOSITION 17. $\sim_\theta$ *is a congruence for $\mu\text{PPA}(A)$.*

PROOF. We shall prove by induction on the size of $\Gamma \vdash P : \mu\text{PPA}(A)$ that if $\Delta \vdash \sigma \sim_\theta \sigma' : \Gamma$ then $\Delta \vdash P[\sigma] \sim_\theta P[\sigma'] : \mu\text{PPA}(A)$. The only case different from Proposition 5 is if $P = \mu x . Q$. Let y be a fresh variable. Then:

$$\Delta \vdash \theta(\mu x . Q)[\sigma] \sim \theta Q[x := \mu x . Q][\sigma]$$
$$\sim \theta Q[x := y][y := \mu x . Q][\sigma]$$
$$\sim \theta Q[x := y][\sigma][y := (\mu x . Q)[\sigma]]$$
$$\sim \theta Q[x := y][\sigma][y := \theta(\mu x . Q)[\sigma]] : \mu\text{PPA}(A)$$

Similarly:

$$\Delta \vdash \theta(\mu x \,.\, Q)[\sigma'] \sim \theta Q[x := y][\sigma'][y := \theta(\mu x \,.\, Q)[\sigma']] : \mu\mathrm{PPA}(A)$$

so:

$$\begin{aligned} \Delta \vdash \theta(\mu x \,.\, Q)[\sigma] &\sim \mu y \,.\, (\theta Q[x := y][\sigma]) \\ &\sim \mu y \,.\, (\theta Q[x := y][\sigma']) \\ &\sim \theta(\mu x \,.\, Q)[\sigma] : \mu\mathrm{PPA}(A) \end{aligned}$$

Thus $\Delta \vdash \mu x \,.\, Q[\sigma] \sim_\theta \mu x \,.\, Q[\sigma'] : \mu\mathrm{PPA}(A)$. $\qquad\square$

6 Translating other prioritized calculi into PPA

In the appendices, we provide a translation into PPA of CLEAVELAND and HEN-NESSY's prioritized CCS, a subset of BAETEN, BERGSTRA and KLOP's ACP_θ, and a variant of HANSSON and ORAVA's PC. For each process P in the source language, we provide a translation $[\![P]\!]$ into PPA, and show that $P \sim [\![P]\!]$. In particular, for each constructor op of the source language, op $x \sim [\![\mathrm{op}\,x]\!]$ and so each of the constructors can be translated into PPA. Thus, the constructors of PPA are at least as expressive as the constructors of PCCS, ACP_θ and PC.

Unfortunately, not all of the constructors of each language can be translated into PPA, since not all of them respect prioritized bisimulation. For example, ACP_θ has a constructor $p \,|\, q$, with the operational semantics:

$$\frac{p \xrightarrow{a} p' \quad q \xrightarrow{b} q'}{p \,|\, q \xrightarrow{c} p' \,\|\, q'} \; [a \,|\, b = c]$$

where $| : A_\delta \to A_\delta \to A_\delta$. Then if $a < b$, $a \,|\, a = a$ and $a \,|\, b = \delta$ then:

$$a + b \sim_\theta b$$

but:

$$(a + b) \,|\, a \sim_\theta a \not\sim_\theta \delta \sim_\theta b \,|\, a$$

Thus $|$ does not preserve prioritized bisimulation, which is one reason why unprioritized bisimulation is investigated by BAETEN, BERGSTRA and KLOP. Similarly, in order to make sure that prioritized bisimulation is respected by $p \,\|\, q$ and $\partial_H p$, we have to put the following restrictions on $|$ and H:

- If $a < c$ then $a \,|\, b < c$.
- H is $<$-downwards closed.

HANSSON and ORAVA (1992, Proposition 5) claim that their variant of prioritized bisimulation $\sim_\pi$ is a congruence for their language PC. Unfortunately, it is not, since if we define:

$$P = a(1).d(1) \,|_{\{a\}}\, a(1) + b(3) \,|_{\{b,d\}}\, b(3) + c(7) + d(1)$$

$$Q = a(1).e(1) \mid_{\{a\}} a(1) + b(3) \mid_{\{b,e\}} b(3) + c(7) + e(1)$$
$$R = a(1).d(1) \mid_{\{a\}} a(1) + b(0) \mid_{\{b,d\}} b(0) + c(7) + d(1)$$

then $P \setminus \{a,b\} \sim_\pi Q \setminus \{a,b\}$ but $P \setminus \{a,b\} \setminus \{c\} \not\sim_\pi Q \setminus \{a,b\} \setminus \{c\}$. Prioritized bisimulation $\sim_\theta$ is also not a congruence, since $P \setminus \{b,c\} \sim_\theta R \setminus \{b,c\}$ but $P \setminus \{b,c\} \setminus \{a\}_\theta R \setminus \{b,c\} \setminus \{a\}$.

As discussed in Appendix C, this is because the priority 'order' implicitly used by HANSSON and ORAVA is not transitive. For example, in $P \setminus \{a,b,c\}$, the internal a action is lower priority than the internal b action, which is lower priority than the internal c action, but the internal a action is incomparable with the internal c action, since they are performed by concurrent processes.

In addition, HANSSON and ORAVA allow unguarded recursion, which is problematic. If we define:

$$A \stackrel{\text{def}}{=} a(0).\mathbf{0} + (a(1) \mid_{\{a\}} A)$$

Then for any n, A can perform an a with priority n, and so $A \setminus \{a\}$ is not well-defined in HANSSON and ORAVA's transition system.

In Appendix C we propose a different priority relation for PC, which is a partial order, and we can show that the resulting calculus with guarded recursion can be translated into PPA.

The only operators of prioritized CCS, ACP$_\theta$ and PC which cannot be translated into PPA are those which do not respect prioritized bisimulation. This is an example of a tradeoff between the expressivity of a language, and having pleasant mathematical properties.

7   Conclusions

In this paper, we have provided:

- A typed, prioritized process algebra.
- An operational semantics, which provides a notion of prioritized bisimulation.
- A complete axiomatization for prioritized bisimulation.
- A translation of three other prioritized process algebras into PPA.

It is worth noting that the three translations make heavy use of the type structure of PPA, and it is not obvious whether there would be a similar simple translation into a language without the type structure.

There are a number of open problems still to be resolved:

- Is there a treatment of priority which unifies priority by *action* (dealt with here) and priority by *process* (dealt with, for example, by CAMILLARI (1990))?
- Is there a transition system format for which PPA is completely expressive, in the same way as DE SIMONE (1985) has shown SCCS and the author (1992a)

$$\frac{}{\sigma.t \xrightarrow{\sigma} t} \quad \frac{t \xrightarrow{\sigma} t'}{t+u \xrightarrow{\sigma} t'} \quad \frac{u \xrightarrow{\sigma} u'}{t+u \xrightarrow{\sigma} u'} \quad \frac{t \xrightarrow{\sigma} t'}{t \setminus a \xrightarrow{\sigma} t' \setminus a} \; [\sigma \notin \{a, \overline{a}, \underline{a}, \underline{\overline{a}}\}]$$

$$\frac{t \xrightarrow{\sigma} t'}{t \mid u \xrightarrow{\sigma} t' \mid u} \quad \frac{u \xrightarrow{\sigma} u'}{t \mid u \xrightarrow{\sigma} t \mid u'} \quad \frac{t \xrightarrow{\alpha} t' \quad u \xrightarrow{\overline{\alpha}} u'}{t \mid u \xrightarrow{\tau} t' \mid u'} \quad \frac{t \xrightarrow{\underline{\alpha}} t' \quad u \xrightarrow{\underline{\overline{\alpha}}} u'}{t \mid u \xrightarrow{\underline{\tau}} t' \mid u'}$$

TABLE 5. The operational semantics of PCCS

has shown CSP is completely expressive for *de Simone format*?

- Can the partial order used to model concurrent prioritized actions be unified with the partial order structures investigated by MAZURKIEWICZ (1986), PRATT (1986), WINSKEL (1989) and many others?

The appendices contain detailed comparisons of PPA with three other approaches to priority.

A Translating prioritized CCS into PPA

CLEAVELAND and HENNESSY's (1988) prioritized variant of CCS has actions which are either low priority (written α) or high priority (written $\underline{\alpha}$). There are two silent actions, one of low priority (τ) and one of high priority ($\underline{\tau}$). The only priority law in their axiomatization is:

$$\alpha.P + \underline{\tau}.Q = \underline{\tau}.P$$

More formally, there is a set Λ of labels, with a complement $\overline{}$ such that $\overline{\overline{a}} = a$. The set of actions is $\mathbf{A} = \Lambda_\tau \cup \{\underline{\alpha} \mid \alpha \in \Lambda_\tau\}$, ranged over by σ. The language PCCS is defined:

$$t ::= nil \mid \sigma.t \mid t + t \mid t \mid t \mid t \setminus a \mid x$$

This is given an operational semantics in Table 5, and the prioritized operational semantics is given as:

- If $t \xrightarrow{\alpha} u$ then $t \xmapsto{\underline{\alpha}} u$.
- If $t \xrightarrow{\alpha} u$ and $t \xnrightarrow{\underline{\tau}}$ then $t \xmapsto{\alpha} u$.

To translate PCCS into PPA, we define the strict partial order on $\mathbf{A}$ as the smallest such that:

$$\alpha < \underline{\tau}$$

Then define:

$$Fail = \{(\sigma,\sigma') \mid \overline{\sigma} \neq \sigma'\}$$

$$[\![nil]\!] = \delta \qquad\qquad\qquad \text{flat} \; : \; \partial_{Fail}(\mathbf{A} \,|\, \mathbf{A}) \to \mathbf{A}$$

$$[\![\sigma.t]\!] = \sigma.[\![t]\!] \qquad\qquad \text{flat}(\text{inl}\,\sigma) = \sigma$$

$$[\![t+u]\!] = [\![t]\!] + [\![u]\!] \qquad\qquad \text{flat}(\text{inr}\,\sigma) = \sigma$$

$$[\![t \,|\, u]\!] = \rho_{\text{flat}}\partial_{\text{sync}}([\![t]\!] \,|\, [\![u]\!]) \qquad \text{flat}(a,\overline{a}) = \tau$$

$$[\![t \setminus a]\!] = \rho_{\text{inject}}\partial_{\{a,\overline{a},\underline{a},\underline{\overline{a}}\}}[\![t]\!] \qquad \text{flat}(\underline{a},\underline{\overline{a}}) = \underline{\tau}$$

$$[\![x]\!] = x \qquad\qquad\qquad \text{inject} \; : \; (\partial_H\mathbf{A}) \to \mathbf{A}$$

$$\text{inject}\,\sigma = \sigma$$

Then we can show that the PPA translation models PCCS up to unprioritized bisimulation.

PROPOSITION 18.

1. *If $t \xrightarrow{\sigma} t'$ then $[\![t]\!] \xrightarrow{\sigma} \sim [\![t']\!]$.*
2. *If $[\![t]\!] \xrightarrow{\sigma} P$ then $t \xrightarrow{\sigma} t'$ and $[\![t']\!] \sim P$.*
3. *$(t, \longrightarrow) \sim ([\![t]\!], \longrightarrow)$.*

PROOF.

1. An induction on t.

2. An induction on t.

3. Define:

$$\mathcal{R} = \{(t, [\![t]\!]) \mid t \in \text{PCCS}\}$$

   Then from parts 1 and 2 we can show that $\mathcal{R}$ is a strong bisimulation up to strong bisimulation, and so $t \sim [\![t]\!]$. □

We can show that this translation models PCCS up to prioritized bisimulation.

PROPOSITION 19.

1. *If $t \xmapsto{\sigma} t'$ then $\theta[\![t]\!] \xrightarrow{\sigma} \sim \theta[\![t']\!]$.*
2. *If $\theta[\![t]\!] \xrightarrow{\sigma} P$ then $t \xmapsto{\sigma} t'$ and $\theta[\![t']\!] \sim P$.*
3. *$(t, \longmapsto) \sim (\theta[\![t]\!], \longrightarrow)$.*

PROOF.

1. If $t \xmapsto{\underline{\alpha}} t'$ then $t \xrightarrow{\underline{\alpha}} t'$ so by Proposition 18 $[\![t]\!] \xrightarrow{\underline{\alpha}} \sim [\![t']\!]$. Since $\underline{\alpha}$ is $<$-maximal, $[\![t]\!] \downarrow \underline{\alpha}$, so $\theta[\![t]\!] \xrightarrow{\underline{\alpha}} \sim \theta[\![t']\!]$.

   If $t \xmapsto{\alpha} t'$ then $t \xrightarrow{\alpha} t'$ and $t \not\xrightarrow{\underline{\tau}}$, so by Proposition 18 $[\![t]\!] \xrightarrow{\alpha} \sim [\![t']\!]$. Since $t \not\xrightarrow{\underline{\tau}}$, $[\![t]\!] \downarrow \alpha$, so $\theta[\![t]\!] \xrightarrow{\alpha} \sim \theta[\![t']\!]$.

2. Similar to part 1.

$$\frac{}{\epsilon\surd}\quad\frac{p\surd\quad q\surd}{p.q\surd}\quad\frac{p\surd}{p+q\surd}\quad\frac{q\surd}{p+q\surd}\quad\frac{p\surd\quad q\surd}{p\,\|\,q\surd}\quad\frac{p\surd}{\partial_H p\surd}\quad\frac{p\surd}{\theta p\surd}$$

TABLE 6. The termination of ACP_θ

$$\frac{}{a\xrightarrow{a}\epsilon}\quad\frac{p\xrightarrow{a}p'}{p.q\xrightarrow{a}p'.q}\quad\frac{p\surd\quad q\xrightarrow{a}q'}{p.q\xrightarrow{a}q'}\quad\frac{p\xrightarrow{a}p'}{p+q\xrightarrow{a}p'}\quad\frac{q\xrightarrow{a}q'}{p+q\xrightarrow{a}q'}$$

$$\frac{p\xrightarrow{a}p'}{p\,\|\,q\xrightarrow{a}p'\,\|\,q}\quad\frac{q\xrightarrow{a}q'}{p\,\|\,q\xrightarrow{a}p\,\|\,q'}\quad\frac{p\xrightarrow{a}p'\quad q\xrightarrow{b}q'}{p\,\|\,q\xrightarrow{a|b}p'\,\|\,q'}\ [a\,|\,b\neq\delta]$$

$$\frac{p\xrightarrow{a}p'\quad p\downarrow a}{\theta p\xrightarrow{a}\theta p'}\quad\frac{p\xrightarrow{a}p'}{\partial_H p\xrightarrow{a}\partial_H p'}\ [a\notin H]$$

TABLE 7. The operational semantics of ACP_θ

3. Similar to Proposition 18. $\square$

B   Tanslating prioritized ACP into PPA

BAETEN, BERGSTRA and KLOP's (1986) ACP_θ has actions **A** with a strict partial order $<$ and a strict, associative, commutative communication function $|:\mathbf{A}_\delta\times\mathbf{A}_\delta\to\mathbf{A}_\delta$. A priority law derivable from their axiomatization is:

$$\theta(a.p+b.q)=\theta(b.q)\qquad(a<b)$$

We shall not translate the entire language ACP_θ into PPA, for reasons discussed in Section 6. The subset we shall consider is defined:

$$p::=\delta\mid\epsilon\mid a\mid p.p\mid p+p\mid p\,\|\,p\mid\partial_H p\mid\theta p\mid x$$

where:

- If $a<c$ then $a\,|\,b<c$.
- H is an $<$-downwards-closed set of actions.

This is given an operational semantics in Tables 6 and 7. Then define:

$$[\![\delta]\!]=\delta$$
$$[\![\epsilon]\!]=\epsilon$$
$$[\![a]\!]=a$$
$$[\![p+q]\!]=[\![p]\!]+[\![q]\!]$$
$$[\![p.q]\!]=[\![p]\!].[\![q]\!]$$
$$[\![p\,\|\,q]\!]=\rho_{\text{inject}}\partial_{\{\delta\}}\rho_{\text{comm}}([\![p]\!]\mid[\![q]\!]))$$
$$[\![\partial_H p]\!]=\partial_H[\![p]\!]$$
$$[\![\theta p]\!]=\theta[\![p]\!]$$
$$[\![x]\!]=x$$

$$\text{comm}\ :\ \mathbf{A}\,|\,\mathbf{A}\to\mathbf{A}_\delta$$
$$\text{comm}(\text{inl}\,a)=a$$
$$\text{comm}(\text{inr}\,a)=a$$
$$\text{comm}(a,b)=a\,|\,b$$
$$\text{inject}\ :\ \partial_H\mathbf{A}\to\mathbf{A}$$
$$\text{inject}\,a=a$$

Then we can show that the PPA translation models ACP_θ up to unprioritized bisimulation.

PROPOSITION 20.

1. If $p \xrightarrow{\sigma} p'$ then $[\![p]\!] \xrightarrow{\sigma} \sim [\![p']\!]$.
2. If $[\![p]\!] \xrightarrow{\sigma} P$ then $p \xrightarrow{\sigma} p'$ and $[\![p']\!] \sim P$.
3. $(p, \longrightarrow) \sim ([\![p]\!], \longrightarrow)$.

PROOF. Similar to Propisition 19.   □

C   Tanslating PC into PPA

HANSSON and ORAVA's (1992) Priority Calculus PC assumes a set of actions $\Lambda_\tau = \Lambda \cup \{\tau\}$ and a total order of priorities $(Pri, \leq)$ with a monotone, cancellative, commutative monoid $(Pri, +, 0)$. Their priority law is only between silent actions:

$$\tau(\pi).P + \tau(\pi').Q = \tau(\pi').Q \qquad (\pi < \pi')$$

In addition, priority is only important between actions performed 'on the same processor', for example:

$$\tau(0).P \mid_L \tau(1).Q \not\sim_\theta \tau(1).(\tau(0).P \mid_L Q)$$

since in the LHS, the $\tau(1)$ action does not override the $\tau(0)$ action, since they are performed concurrently. The syntax of PC is:

$$P ::= \mathbf{0} \mid \alpha(\pi).P \mid P + P \mid P \mid_L P \mid P \setminus L \mid x \mid \mu x . P$$

where $L \subseteq \Lambda$. This is given a proof-labelled transition system similar to those of BOUDOL and CASTELLANI (1988). Transitions are of the form $P \xmapsto{(\alpha, \pi, O)} P'$ where $O \subseteq \{l, r, v, h\}^*$ records how the transition was deduced. Let $\mathbf{A} = \Lambda_\tau \times Pri \times \mathbf{P}(\{l, r, v, h\}^*)$. The transition system is given in Table 8, where:

$$O \bullet x = \{\sigma x \mid \sigma \in O\}$$

In a transition $P \xmapsto{(\alpha, \pi, O)} P'$:

- If $\sigma v \in O$ then the transition comes from the LHS of a $+$.
- If $\sigma h \in O$ then the transition comes from the RHS of a $+$.
- If $\sigma l \in O$ then the transition comes from the LHS of a $\mid$.
- If $\sigma r \in O$ then the transition comes from the RHS of a $\mid$.

Note that a set O may have more than one element, where each string names a concurrent component cooperating in the action. For example, a transition $(\alpha, \pi, \{l, r\})$ must come from both the LHS and the RHS of a $\mid$, such as:

$$a(1).\mathbf{0} \mid_{\{a\}} a(1).\mathbf{0} \xmapsto{(a, 2, \{l, r\})} \mathbf{0} \mid_{\{a\}} \mathbf{0}$$

The transition system $\longmapsto$ makes no mention of the order $\leq$. It is introduced by defi ning a priority ordering on $\mathbf{A}$. The priorty ordering introduced implicitly by

$$\overline{\alpha(\pi).P \xrightarrow{(\alpha,\pi,\{\epsilon\})} P}$$

$$\frac{P \xrightarrow{(\alpha,\pi,O)} P'}{P+Q \xrightarrow{(\alpha,\pi,O\bullet h)} P'+\mathbf{0}} \quad \frac{Q \xrightarrow{(\alpha,\pi,O)} Q'}{P+Q \xrightarrow{(\alpha,\pi,O\bullet v)} \mathbf{0}+Q'}$$

$$\frac{P \xrightarrow{(\alpha,\pi,O)} P'}{P|_L Q \xrightarrow{(\alpha,\pi,O\bullet l)} P'|_L Q} \ [\alpha \notin L]$$

$$\frac{Q \xrightarrow{(\alpha,\pi,O)} Q'}{P|_L Q \xrightarrow{(\alpha,\pi,O\bullet r)} P|_L Q'} \ [\alpha \notin L]$$

$$\frac{P \xrightarrow{(a,\pi,O)} P' \quad Q \xrightarrow{(a,\pi',O')} Q'}{P|_L Q \xrightarrow{(a,\pi+\pi',O\bullet l\cup O'\bullet r)} P'|_L Q'} \ [a \in L]$$

$$\frac{P \xrightarrow{(\alpha,\pi,O)} P'}{P\setminus L \xrightarrow{(\alpha,\pi,O)} P'\setminus L} \ [\alpha \notin L]$$

$$\frac{P \xrightarrow{(\alpha,\pi,O)} P'}{P\setminus L \xrightarrow{(\tau,\pi,O)} P'\setminus L} \ [\alpha \in L]$$

TABLE 8. The operational semantics of PC

HANSSON and ORAVA is given:

$$(\tau,\pi,O) \prec (\tau,\pi',O') \quad \text{iff} \quad \pi < \pi' \text{ and Comp}(O,O')$$

where:

- Comp(σ,σ), Comp$(\sigma v, \sigma' h)$ and Comp$(\sigma h, \sigma' v)$.
- if Comp(σ,σ') then Comp$(\sigma l, \sigma' r)$, Comp$(\sigma r, \sigma' l)$ and Comp$(\sigma x, \sigma' x)$.
- Comp(O,O') iff $\exists \sigma \in O, \sigma' \in O'$. Comp$(\sigma,\sigma')$.

Unfortunately, this relation is not transitive, and as a result prioritized bisimulation is not a congruence. For example:

$$(\tau,2,\{ll,vrl\}) \prec (\tau,6,\{hrl,vvr\}) \prec (\tau,7,\{hvr\})$$

but:

$$(\tau,2,\{ll,vrl\}) \nprec (\tau,7,\{hvr\})$$

This counter-example is used in showing the counter-example in Section 6 where:

$$P\setminus\{b,c\} \sim_\theta R\setminus\{b,c\} \quad \text{but} \quad P\setminus\{b,c\}\setminus\{a\} \nsim_\theta R\setminus\{b,c\}\setminus\{a\}$$

Here, we propose an alternative order on **A**, which only allows priorities to be compared when the actions are performed on the same processor. Given $\sigma \in \{l,r,v,h\}^*$ we can define the *processor identifier* of σ to be:

$$\text{pid}(r\sigma) = r\sigma \quad \text{pid}\epsilon = \epsilon \quad \text{pid}(h\sigma) = \text{pid}\,\sigma$$
$$\text{pid}(l\sigma) = l\sigma \quad \quad \quad \text{pid}(v\sigma) = \text{pid}\,\sigma$$

Then we will define $(\tau,\pi,O) < (\tau,\pi',O')$ iff $\pi < \pi'$ and the processors of O are

a subset of the processors of O':

$$(\tau, \pi, O) < (\tau, \pi', O') \quad \text{iff} \quad \pi < \pi' \text{ and } \text{pid}[O] \subseteq \text{pid}[O']$$

For example, this means that:

$$\tau(0).P + \tau(1).Q \sim_\theta \mathbf{0} + \tau(1).Q$$

since $(\tau, 0, \{v\}) < (\tau, 1, \{h\})$ but:

$$(\tau(0).P \mid_L \mathbf{0}) + \tau(1).R \not\sim_\theta \mathbf{0} + \tau(1).R$$

since $(\tau, 0, \{lv\}) \not< (\tau, 1, \{h\})$. This priority relation seems to capture the motivation of only allowing non-concurrent actions to be compared, whilst making $\sim_\theta$ a congruence. We can use this relation to define the transition system $\longrightarrow$ on PC:

$$\frac{P \xmapsto{(\alpha, \pi, O)} P' \quad P \downarrow (\alpha, \pi, O)}{P \xrightarrow{(\alpha, \pi, O)} P'}$$

Then define:

$$\llbracket \mathbf{0} \rrbracket = \delta$$
$$\llbracket \alpha(\pi).P \rrbracket = (\alpha, \pi, \{\varepsilon\}).\llbracket P \rrbracket$$
$$\llbracket P + Q \rrbracket = \rho_{\text{snoc}_v}\llbracket P \rrbracket + \rho_{\text{snoc}_h}\llbracket Q \rrbracket$$
$$\llbracket P \mid_L Q \rrbracket = \rho_{\text{flat}_L} \partial_{Fail_L}((\rho_{\text{snoc}_l}\llbracket P \rrbracket) \mid (\rho_{\text{snoc}_r}\llbracket Q \rrbracket))$$
$$\llbracket P \setminus L \rrbracket = \rho_{\text{hide}_L}\llbracket P \rrbracket$$
$$\llbracket x \rrbracket = x$$
$$\llbracket \mu x.P \rrbracket = \mu x.\llbracket P \rrbracket$$

where:

$$Fail_L = \{\text{inl}(\alpha, \pi, O) \mid \alpha \in L\}$$
$$\cup \{\text{inr}(\alpha, \pi, O) \mid \alpha \in L\}$$
$$\cup \{((\alpha, \pi, O), (\beta, \pi', O') \mid \alpha \neq \beta\}$$
$$\text{flat}_L : \partial_{Fail_L}\mathbf{A} \to \mathbf{A}$$
$$\text{flat}_L(\text{inl}(\alpha, \pi, O)) = (\alpha, \pi, O)$$
$$\text{flat}_L(\text{inr}(\alpha, \pi, O)) = (\alpha, \pi, O)$$
$$\text{flat}_L((\alpha, \pi, O), (\alpha, \pi', O')) = (\alpha, \pi + \pi', O \cup O')$$
$$\text{snoc}_x : \mathbf{A} \to \mathbf{A}$$
$$\text{snoc}_x(\alpha, \pi, O) = (\alpha, \pi, O \bullet x)$$
$$\text{hide}_L : \mathbf{A} \to \mathbf{A}$$
$$\text{hide}_L(\alpha, \pi, O) = \begin{cases} (\tau, \pi, O) & \text{if } \alpha \in L \\ (\alpha, \pi, O) & \text{otherwise} \end{cases}$$

Then we can show that the PPA translation models PC up to unprioritized bisimulation.

PROPOSITION 21.

1. If $P \xmapsto{a} P'$ then $[\![P]\!] \xrightarrow{a} \sim [\![P']\!]$.
2. If $[\![P]\!] \xrightarrow{a} Q$ then $P \xmapsto{a} P'$ and $[\![P']\!] \sim Q$.
3. $(P, \longmapsto) \sim ([\![P]\!], \longrightarrow)$.

PROOF. Similar to Propisition 19. □

We can also show that the PPA translation models PC up to prioritized bisimulation.

PROPOSITION 22. $(P, \longrightarrow) \sim (\theta[\![P]\!], \longrightarrow)$.

PROOF. Follows from the definition of $(P, \longrightarrow)$ and the operational semantics of θp. □

References

BAETEN, J. C. M., BERGSTRA, J. A., and KLOP, J. W. (1986). Syntax and defining equations for an interrupt mechanism in process algebra. *Fund. Inform.*, 9(2):127–168.

BAETEN, J. C. M. and WEIJLAND, W. P. (1990). *Process Algebra*. Cambridge University Press.

BOUDOL, G. and CASTELLANI, I. (1988). A non-interleaving semantics for CCS based on proved transitions. *Fund. Inform.*, XI:433–452.

CAMILLERI, J. (1990). *Priority in Process Calculi*. PhD thesis, Cambridge University.

CLEAVELAND, R. and HENNESSY, M. (1988). Priorities in process algebra. In *Proc. LICS 88*. The Computer Society.

DE SIMONE, R. (1985). Higher-level synchronising devices in Meije-SCCS. *Theoret. Comput. Sci.*, 37:245–267.

GROOTE, J. F. (1989). Transition system specifications with negative premises. Report CS-R8950, CWI, Amsterdam.

GROOTE, J. F. and VAANDRAGER, F. W. (1989). Structured operational semantics and bisimulation as a congruence. In *Proc. ICALP 89*, pages 423–438. Springer-Verlag. LNCS 372.

HANSSON, H. and ORAVA, F. (1992). A process calculus with incomparable priorities. In *Proc. NAPAW '92*. Department of Computer Science, Johns Hopkins University.

JEFFREY, A. (1992a). CSP is completely expressive. Technical report 2/92, University of Sussex.

JEFFREY, A. (1992b). *Observation Spaces and Timed Processes*. D.Phil thesis, Oxford University. Available as technical report PMG-R64 from the Programming Methodology Group, Chalmers University.

MAZURKIEWICZ, A. (1986). Trace theory. In BRAUER, W., REISIG, W., and ROZENBERG, G., editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*. Springer-Verlag. LNCS 255.

MILNER, R. (1989). *Communication and Concurrency*. Prentice-Hall.

PRATT, V. (1986). Modelling concurrency with partial orders. *Internat. J. Parallel Programming*, 15(1):33–71.

SCHNEIDER, S. A. (1990). *Communication and Correctness in Real-time Distributed Computing*. D.Phil. thesis, Oxford University.

WINSKEL, G. (1984). Categories of models for concurrency. In BROOKES, S. D., ROSCOE, A. W., and WINSKEL, G., editors, *Proc. Seminar on Semantics of Concurrency*, pages 246–267. Springer-Verlag.

WINSKEL, G. (1989). An introduction to event structures. In DE BAKKER, J., DE ROEVER, W., and ROZENBERG, G., editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*. Springer-Verlag. LNCS 354.