iccp@doc.ic.ac.uk
Department of Computing
Imperial College
London SW7 2BZ

CCS WITH PRIORITY GUARDS

IAIN PHILLIPS

Abstract. It has long been recognised that ordinary process algebra has difficulty dealing with actions of different priority, such as for instance an interrupt action of high priority. Various solutions have been proposed. We introduce a new approach, involving the addition of "priority guards" to the summation operator of Milner's process calculus CCS. In our approach, priority is *unstratified*, meaning that actions are not assigned fixed levels, so that the same action can have different priority depending where it appears in a program. An important feature is that, unlike in other unstratified accounts of priority in CCS (such as that of Camilleri and Winskel), we can treat inputs and outputs symmetrically. We introduce the new calculus, give examples, develop its theory (including bisimulation, equational laws and logics), and compare it with existing approaches.

1. Introduction

It has long been recognised that ordinary process algebra [Mil99, Hoa85, BK84] has difficulty dealing with actions of different priority, such as for instance an interrupt action of high priority. Various authors have suggested how to add priority to process languages [BBK86, CH90, HO92, Fid93, Jef93, Pra94, CW95, SS96]. We introduce a new approach, involving the addition of "priority guards" to the summation operator of Milner's process calculus CCS. In our approach, priority is *unstratified*, meaning that actions are not assigned fixed levels, so that the same action can have different priority depending where it appears in a program. We shall see that existing accounts of priority in CCS are either stratified [CH90], or else they impose a distinction between outputs and inputs, whereby prioritised choice is only made on inputs [CW95, CLN00]. This goes against the spirit of CCS, where inputs and outputs are treated symmetrically, and we contend that it is unnecessary. We introduce the new calculus, give examples, develop its theory (including bisimulation, equational laws and logics), and compare it with existing approaches.

We start with the idea of priority. We assume some familiarity with CCS notation [Mil99]. Consider the CCS process $a.0 + b.0$. The actions a and b have equal status. Which of them engages in communication depends on whether the environment is offering the complementary actions $\bar{a}$ or $\bar{b}$. By "environment" we mean whatever processes may be placed in parallel with $a.0 + b.0$. If the environment offers both, then the choice is nondeterministic. We would like some means to favour a over b, say, so that if the environment offers both, then only a can happen. This would be useful if, for instance, a was an interrupt action. We need something more sophisticated than simply removing b altogether, since, if a cannot communicate, it should not stop b from doing so. This brief analysis points to two features of priority: (1) Priority removes ("preempts") certain possibilities that would have

Key words and phrases. Process algebra, CCS, bisimulation, Hennessy-Milner logic, priority, interrupt.

Partially funded by EPSRC grant GR/K54663.

existed without priority. Thus if a can communicate then b is preempted. (2) Parallel composition plays a crucial rôle in priority in CCS.

We now explain our basic idea. Let P be a process, let a be an action, and let U be some set of actions. Then we can form a new process $U : a.P$, which behaves like $a.P$, except that the initial action a is conditional on the environment not offering actions in $\bar{U}$, the CCS "complement" of U. We call U a *priority guard* in $U : a.P$. All actions in U have priority over a at this point in the computation. We call our calculus CPG (for CCS with Priority Guards).

As a simple example, if we have a CCS process $a.P + b.Q$ and we wish to give a priority over b in the choice, we add a priority guard to get $a.P + a : b.Q$ (we omit the set braces around a). Priority is specific to this choice, since the guard affects only the initial b, and not any further occurrences of b there may be in Q.

Let us see how this example is handled in two existing approaches to priority. Camilleri and Winskel proposed a priority choice operator [CW95]. In their notation the example becomes $a.P + \!\rangle b.Q$. Cleaveland and Hennessy [CH90] add new higher priority actions to CCS. They would write our example as $\underline{a}.P + b.Q$ (high priority actions are underlined).

In Cleaveland and Hennessy's stratified calculus, actions have fixed priority levels, and $\underline{a}$ always has priority over b, not just in the choice under consideration, but throughout the system. Furthermore if a third action c is present in the system, then $\underline{a}$ also has priority over it. The stratification means that only actions at the same priority level can communicate. In this paper we are interested in an unstratified approach, and so our starting point of reference is Camilleri and Winskel's work. They make the priority of a over b totally specific to the particular choice, so that b might have priority over a elsewhere in the same program. However they limit themselves so prioritised choice on *input* actions—only normal choice is allowed on output actions. In our proposal, inputs and outputs are treated equally, which is more in the spirit of CCS, where they are simply dual. We shall see that there is an interesting contrast between our system and theirs in the treatment of preemption. We also note that Camilleri and Winskel's work does not hide silent (τ) actions, and has never been extended to do so, as far as we are aware. Finally, we contend that our system is more tractable, both in terms of operational semantics and equational theory. We shall discuss this further in Section 2.

To end this section, we give an example, involving handling of hidden actions and the scoping of priority. We wish to program a simple interrupt. Let P be a system which consists of two processes A, B in parallel which perform actions a, b respectively, while communicating internally to keep in step with each other. P also has an interrupt process I which shuts down A and B when the interrupt signal *int* is received.

$$P \stackrel{\mathrm{df}}{=} \mathrm{new}\ \mathsf{mid}, \mathsf{int}_A, \mathsf{int}_B\ (A|B|I) \qquad A \stackrel{\mathrm{df}}{=} \mathsf{int}_A : a.\overline{\mathsf{mid}}.A + \mathsf{int}_A.0$$
$$I \stackrel{\mathrm{df}}{=} \mathsf{int}.(\overline{\mathsf{int}}_A.\overline{\mathsf{int}}_B.0 + \overline{\mathsf{int}}_B.\overline{\mathsf{int}}_A.0) \quad B \stackrel{\mathrm{df}}{=} \mathsf{int}_B : b.\mathsf{mid}.B + \mathsf{int}_B.0$$

Without the priority guards in A and B, P could receive an *int* and yet A and B could continue with a and b. Actions int_A, int_B have priority over a, b, respectively. This only applies within the scope of the restriction. We can apply the usual techniques of CCS (including removing τ actions) and get

$$P = a.P_1 + b.P_2 + \mathsf{int}.0 \quad P_1 = b.P + \mathsf{int}.0 \quad P_2 = a.P + \mathsf{int}.0$$

which is what we wanted. We consider this example more precisely in Section 8.

It is worth noting that if we merely wanted to interrupt A, we could have simply defined $A \stackrel{\mathrm{df}}{=} \mathsf{int}_A : a.\overline{\mathsf{mid}}.A$. As soon as int occurs, I offers $\overline{\mathsf{int}}_A$, and A cannot perform a according to our rules. The choice of $\mathsf{int}_A.0$ in A gives us a way of gracefully recovering from the interrupt.

The rest of the paper is organised as follows: First we compare our approach with related work (Section 2). Next we define the language of processes (Section 3). Then we look at reactions (Section 4) and labelled transitions (Section 5). We then look at bisimulation and equational theories for both the strong (Section 6) and weak cases (Section 7). We then return to our interrupt example (Section 8), before examining logics for priority (Section 9). The paper is completed with some brief conclusions.

2. Related Work

We refer the reader to [CLN00] for discussion of the many approaches taken by other authors to priority. Here we restrict ourselves to comparison of our work with that of Camilleri and Winskel [CW95] (referred to as CW for short) and Cleaveland, Lüttgen and Natarajan [CLN00] (CLN for short).

As we have seen, CW's CCS with a prioritised choice operator $P{+\rangle}Q$ allows priority to be decided in a way which is specific to each choice in a system. The idea of a priority choice between processes is interesting and natural. The authors present an operational semantics via a labelled transition relation, and define a bisimulation-based equivalence. They also give an axiomatisation of this equivalence which is complete for finite processes (i.e. those not using recursion). However they do not show how to hide the τ-actions resulting from communications.

The CW transition relation is parametrised on a set of output actions R. Thus $\vdash_R P \xrightarrow{\alpha} P'$ means that, in an environment which is ready to perform precisely the actions R, the process P can perform an action α to become P'. For example, $\vdash_R a.0{+\rangle}b.0 \xrightarrow{a} 0$ (any R), while $\vdash_R a.0{+\rangle}b.0 \xrightarrow{b} 0$ provided $\bar{a} \notin R$. We have adapted the idea of parametrisation on the environment for our labelled transition system for CPG. For us $P \xrightarrow{\alpha}_U P'$ means that, in an environment which offers no action in the set $\bar{U}$ (in our parlance, *eschews* U), process P can perform α to become P'. Our most basic rule is essentially $U : a.P \xrightarrow{a}_U P$, provided $a \notin U$. We feel that we have obtained a satisfying unity between syntax and transition relation.

There is a difference in expressiveness between CPG and CW's calculus, in that the latter cannot express cycles of priority, whereas we can in CPG. CW consider the paradoxical example

$$\mathrm{new}\ a, b\ ((a.0{+\rangle}\bar{b}.0)|(b.0{+\rangle}\bar{a}.0))$$

The problem is that there is a circularity, with a having priority over b, as well as vice versa. Can the system act? They decide to sidestep this question by breaking the symmetry in CCS between inputs and outputs, and only allowing prioritised choice on input actions. We feel that this complicates the syntax and operational semantics, and should not be necessary. In our approach the example is admitted, though it results in a deadlock. We consider this example again at the end of Section 5.

Another reason why CW disallow priority choice on output actions is to assist in obtaining the normal form they use for proving the completeness of their equational laws for finite processes. However this normal form is still quite complicated (consisting of a sum of priority sums of sums). In our calculus CPG we have only one form of choice, and so completeness is technically simpler.

We now turn to CLN's work. In their basic approach [CLN00], which is derived from earlier work of Cleaveland and Hennessy [CH90], actions have priority levels. Mostly they consider just two levels—ordinary actions and higher priority, underlined actions. Only actions at the same level of priority can communicate, which is really quite restrictive when one considers that two actions which are intended to communicate may have quite different priorities within their respective subsystems. The resulting silent actions have preemptive power over all actions of lower priority, i.e. no action of lower priority, whether visible or invisible, can take place in the presence of a higher priority τ-action. The authors present both strong and weak bisimulation-based equivalences (drawing on [NCCC94]), and axiomatise these for finite processes.

In our unstratified calculus CPG, by contrast, actions do not have priority levels—each priority guard operates independently. This is in the spirit of [CW95].

Even disregarding the issue of priority levels, there is a difference between preemption in [CH90, CLN00] and in CPG, since in CPG preemption is done entirely by the environment. Consider the process $a.0 + \underline{\tau}.0$. Here the underlining indicates that $\underline{\tau}$ is a high priority reaction. According to Cleaveland and Hennessy, this process cannot perform a, since it is preempted by $\underline{\tau}$. However if we translate $a.0 + \underline{\tau}.0$ into CPG as $U : a.0 + \tau.0$ (where the set of actions U, with $a \notin U$, is chosen to be as large as necessary), we find that by contrast $U : a.0 + \tau.0 \xrightarrow{a}_U 0$. So the a is not preempted, but only downgraded. This is because performance of a depends on the environment eschewing U. Another example illustrates the opposite effect: For Cleaveland and Hennessy, $a.0|\underline{b}.0 \xrightarrow{a} \underline{b}.0$, but the translation $U : a.0|\underline{b}.0$ (with $\underline{\bar{b}} \in U$) cannot perform a at all, since the environment (in the form of $\underline{b}.0$) is offering an action which preempts a. Here we see that, in CPG, a high priority action can preempt a low priority action in a parallel composition, even without being able to engage in a reaction.

This difference in the handling of preemption means that there is no obvious translation of CPG into the framework of CLN, or vice versa.

The development in [CLN00] goes far beyond the basic Cleveland and Hennessy calculus. They consider distributed priorities, where preemption is decided locally rather than globally. They motivate this by the example of an application which fetches data from two memory benches alternately. In CCS this can be modelled as

$$\mathsf{Appl} \stackrel{\mathrm{df}}{=} \overline{\mathsf{fetch}_1}.\overline{\mathsf{fetch}_2}.\mathsf{Appl}$$

These benches are also connected to a direct-memory-access (DMA) controller. This DMA access should have lower priority than the fetch access by the application. However a straightforward assignment of high priority to application access and low priority to DMA access fails, since one or other of the fetches is always enabled, so that DMA access never takes place.

Their example can be encompassed easily in our unstratified approach. Define

$$\begin{aligned} \mathsf{Bench}_i &\stackrel{\mathrm{df}}{=} \mathsf{fetch}_i.\mathsf{Bench}_i.0 + \mathsf{fetch}_i : \mathsf{dma}.\mathsf{Bench}_i \quad (i = 1, 2) \\ \mathsf{Sys} &\stackrel{\mathrm{df}}{=} \mathrm{new}\ \mathsf{fetch}_1, \mathsf{fetch}_2\ (\mathsf{Appl}|\mathsf{Bench}_1|\mathsf{Bench}_2) \end{aligned}$$

Then Sys has the desired behaviour, since one or other dma action can always take place. By the methods to be used in Section 8, we can show that $\mathsf{Sys} = P$, where $P \stackrel{\mathrm{df}}{=} \mathsf{dma}.P$.

The next step in [CLN00] is to consider extending the distributed priority calculus to allow communication between actions at different levels. The authors identify a problem with associativity of parallel composition. Consider the system

$$(a.0 + \underline{b}.0)|(\bar{b}.0 + \underline{c}.0)|\underline{\bar{c}}.0$$

where communication is allowed between complementary actions at different levels. If this associates to the left, then a is preempted by b; however if it associates to the right then b is preempted by c, and so a is not preempted. A similar problem is encountered when extending the distributed calculus to allow more than two levels.

CLN's proposed solution is to follow CW by only allowing priorities to be resolved between *input* actions, while treating all output actions as having equal priority. We have already mentioned our reservations about this. Nevertheless the distinction between inputs and outputs gives a workable "mixed-level" calculus (distributed, multi-level, with communication between different levels). It is particularly nice that CLN show that the CW calculus can be translated faithfully and naturally into this mixed-level calculus. This shows that the underlying model of preemption is the same in both cases, and, apparently, different from that of CPG.

It is striking that both CW and the mixed-level calculus of CLN adopt the same syntactic restriction on inputs and outputs, and also that only *strong* equivalence (τ actions not hidden) is presented for the mixed-level calculus. We shall present a weak equivalence for CPG.

3. The language CPG

We shall denote our augmentation of CCS with priority guards by *CPG* (CCS with Priority Guards). First we define the actions of CPG. In ordinary CCS [Mil99, Part I] there is a set of *names* $\mathcal{N}$ and a disjoint set of *co-names* $\bar{\mathcal{N}}$, together with a single silent action τ. To these ordinary names $\mathcal{N}$ we shall add a new disjoint set of names Π and a dual set $\bar{\Pi}$. These are the actions which can be used in priority guards; they can also be used in the ordinary way. They need to be kept separate from ordinary actions, since we have to be careful with them in reasoning compositionally about processes.

To see why we take this approach, consider the law $P = \tau.P$, which is valid for CCS processes. In CPG, if a can be a priority guard then $a \neq \tau.a$ since there is a context in which the two sides behave differently. Indeed, $a|\bar{a}:b$ cannot perform b (since, as we shall see, b is preempted by the offer of a), whereas $\tau.a|\bar{a}:b$ can perform b initially, as a is not offered until τ has occurred. However if we know that a is an ordinary name then we do have $a = \tau.a$. So we can retain CCS reasoning when processes only involve ordinary names.

We define $\mathsf{Ord} = \mathcal{N} \cup \bar{\mathcal{N}}$, $\mathsf{Pri} = \Pi \cup \bar{\Pi}$, $\mathsf{Vis} = \mathsf{Ord} \cup \mathsf{Pri}$ and $\mathsf{Act} = \mathsf{Vis} \cup \{\tau\}$. We let $u, v, \ldots$ range over Pri, $a, b, \ldots$ over Vis and $\alpha, \beta, \ldots$ over Act. Also $S, T, \ldots$ range over subsets of Vis, and $U, V \ldots$ over subsets of Pri. If $S \subseteq \mathsf{Vis}$, let $\bar{S}$ denote $\{\bar{a} : a \in S\}$, where if $\bar{a} \in \bar{\mathcal{N}} \cup \bar{\Pi}$ then $\bar{\bar{a}} = a$.

Now we define processes:

Definition 3.1. (cf [Mil99, Definition 4.1]) $\mathcal{P}$ is the smallest set such that whenever P, P_i are processes then $\mathcal{P}$ contains

(1)	$\sum_{i \in I} S_i : \alpha_i.P_i$	guarded summation (I finite)
(2)	$P_1 \| P_2$	parallel composition
(3)	$\mathrm{new}\, a\, P$	restriction
(4)	$A\langle a_1, .., a_n\rangle$	identifier

$\mathcal{P}$ is ranged over by $P, Q, R, \ldots$ We let $M, N, \ldots$ range over (guarded) summations. We assume that each identifier $A\langle b_1, .., b_n\rangle$ comes with a defining equation $A(a_1, .., a_n) \stackrel{\mathrm{df}}{=} P$, where P is a process whose free names are drawn from $a_1, .., a_n$. We will tend to abbreviate $a_1, .., a_n$ by $\vec{a}$. We write the empty guarded summation as 0. It is assumed that the order in a summation is immaterial. We abbreviate $\emptyset : \alpha$ by α. Definition 3.1 is much as in ordinary CCS except for the priority guards

S_i. The meaning of the priority guard $S\!:\!\alpha$ is that α can only be performed if the environment does not offer any action in $\bar{S} \cap \mathsf{Pri}$. Clearly, any names in $S - \mathsf{Pri}$ have no effect as guards, and can be eliminated without changing the behaviour of a process. We allow them to occur in the syntax, since otherwise we could not freely instantiate the parameters in an identifier. We will allow ourselves to write $u : \alpha$ instead of $\{u\} : \alpha$. Restriction is a variable-binding operator, and we write $\mathsf{fn}(P)$ for the *free names* of P, defined as follows:

Definition 3.2. By induction on $P \in \mathcal{P}$:

1. $\mathsf{fn}(\sum_{i\in I} S_i\!:\!\alpha_i.P_i) = \{n \in \mathcal{N} \cup \Pi : \exists i \in I. n \in S_i \cup \{\alpha_i\} \vee \bar{n} \in S_i \cup \{\alpha_i\} \vee n \in \mathsf{fn}(P_i)\}$
2. $\mathsf{fn}(P_1|P_2) = \mathsf{fn}(P_1) \cup \mathsf{fn}(P_2)$
3. $\mathsf{fn}(\text{new } a\ P) = \mathsf{fn}(P) - \{a\}$
4. $\mathsf{fn}(A\langle\vec{b}\rangle) = \mathsf{fn}(\{\overrightarrow{b}/\overrightarrow{a}\}P) \quad \text{if } A(\overrightarrow{a}) \stackrel{\text{df}}{=} P$

Two sublanguages of CPG are of interest:

Definition 3.3. Let $\mathcal{P}_{\mathsf{Ord}}$ be the sublanguage of *ordinary* processes generated as in Definition 3.1 except that all names are drawn from Ord (i.e. we effectively take $\Pi = \emptyset$ and $S_i = \emptyset$ in clause (1)). Let $\mathcal{P}_{\mathsf{Ug}}$ be the sublanguage of *unguarded* processes generated as in Definition 3.1 except that all priority guards are empty (i.e. $S_i = \emptyset$ in clause (1)).

Clearly $\mathcal{P}_{\mathsf{Ord}} \subseteq \mathcal{P}_{\mathsf{Ug}} \subseteq \mathcal{P}$. Note that $\mathcal{P}_{\mathsf{Ord}}$ is effectively normal CCS.

4. Offers and Reaction

Structural congruence is the most basic equivalence on processes, which facilitates reaction by bringing the subprocesses which are to react with each other into juxtaposition. It is defined as for CCS:

Definition 4.1. (cf [Mil99, Definition 4.7]) *Structural congruence*, written $\equiv$, is the congruence on $\mathcal{P}$ generated by the following equations:

1. Change of bound names (alpha-conversion)
2. Reordering of terms in a summation
3. $P|0 \equiv P$, $P|Q \equiv Q|P$, $P|(Q|R) \equiv (P|Q)|R$
4. $\text{new } a\ (P|Q) \equiv P|\text{new } a\ Q \quad \text{if } a \notin \mathsf{fn}(P)$;
   $\text{new } a\ 0 \equiv 0$, $\text{new } a \text{ new } b\ P \equiv \text{new } b \text{ new } a\ P$
5. $A\langle\vec{b}\rangle \equiv \{\overrightarrow{b}/\overrightarrow{a}\}P \quad \text{if } A(\overrightarrow{a}) \stackrel{\text{df}}{=} P$

Recall that a guarded action $S\!:\!a$ is conditional on other processes in the environment not offering actions in $\bar{S} \cap \mathsf{Pri}$. Before defining reaction we must define what it means for the environment to offer an action. We define for each process P the set $\mathsf{off}(P) \subseteq \mathsf{Pri}$ of "higher priority" actions "offered" by P. Note that the offers of a process do not depend on any guards S it may contain.

Definition 4.2. By induction on $P \in \mathcal{P}$:

1. $\mathsf{off}(\sum_{i\in I} S_i\!:\!\alpha_i.P_i) = \{\alpha_i :\ i \in I, \alpha_i \in \mathsf{Pri}, \alpha_i \notin S_i\}$
2. $\mathsf{off}(P_1|P_2) = \mathsf{off}(P_1) \cup \mathsf{off}(P_2)$
3. $\mathsf{off}(\text{new } a\ P) = \mathsf{off}(P) - \{a, \overline{a}\}$
4. $\mathsf{off}(A\langle\vec{b}\rangle) = \mathsf{off}(\{\overrightarrow{b}/\overrightarrow{a}\}P) \text{ if } A(\overrightarrow{a}) \stackrel{\text{df}}{=} P$

In item 1 the reason that we insist $\alpha_i \notin S_i$ is that we want to equate a process such as $u : u$ with 0, since $u : u$ can never engage in a reaction. Note that if $P \in \mathcal{P}_{\mathsf{Ord}}$ then $\mathsf{off}(P) = \emptyset$.

Proposition 4.3. *For any $P \in \mathcal{P}$:*

1. $\mathsf{fn}(P)$ is finite
2. $\mathsf{off}(P) \subseteq \mathsf{fn}(P) \cap \mathsf{Pri}$
3. If $P \equiv Q$ then $\mathsf{fn}(P) = \mathsf{fn}(Q)$ and $\mathsf{off}(P) = \mathsf{off}(Q)$. □

In CPG, a reaction can be conditional on offers from the environment. Consider $u : b.0|\bar{b}.0$. This can react by communication on b, $\bar{b}$. However b is guarded by u, and so the reaction is conditional on the environment not offering $\bar{u}$. We reflect this by letting reaction be parametrised on sets of actions $U \subseteq \mathsf{Pri}$. The intended meaning of $P \rightarrow_U P'$ is that P can react on its own, as long as the environment does not offer $\bar{u}$ for any $u \in U$. Notice that the offers of P to the environment are immaterial here, as are any guarding sets in the environment.

The environment plays both a *positive* and a *negative* role when participating with a process in reaction: Consider for example $u : a.0$. In order to participate in a reaction, this process requires the environment to perform a (positive) and not to offer u (negative). We shall say that the environment should *eschew* u.

Definition 4.4. Let $P \in \mathcal{P}$ and let $S \subseteq \mathsf{Act}$ be finite. P *eschews* S (written $P\,\mathsf{eschews}\,S$) iff $\mathsf{off}(P) \cap \bar{S} = \emptyset$.

Proposition 4.5. *Let $P \in \mathcal{P}$ and let $S \subseteq \mathsf{Act}$ be finite.*

1. If $P\,\mathsf{eschews}\,S$ and $T \subseteq S$ then $P\,\mathsf{eschews}\,T$.
2. If $P\,\mathsf{eschews}\,S$ and $Q\,\mathsf{eschews}\,S$ then $P|Q\,\mathsf{eschews}\,S$.

Proof. Immediate. □

Definition 4.6. (cf [Mil99, Definition 4.13]) The *reaction relation* on $\mathcal{P}$ is the smallest relation $\rightarrow$ on $\mathcal{P} \times \wp(\mathsf{Pri}) \times \mathcal{P}$ generated by the following rules:

$$S : \tau.P + M \rightarrow_{S \cap \mathsf{Pri}} P$$

$$(S : a.P + M)|(T : \bar{a}.Q + N) \rightarrow_{(S \cup T) \cap \mathsf{Pri}} P|Q$$

provided $S : a.P + M\,\mathsf{eschews}\,T$ and $T : \bar{a}.Q + N\,\mathsf{eschews}\,S$

$$\frac{P \rightarrow_U P'}{P|Q \rightarrow_U P'|Q}$$

provided $Q\,\mathsf{eschews}\,U$

$$\frac{P \rightarrow_U P'}{\text{new } a\, P \rightarrow_{U - \{a, \bar{a}\}} \text{new } a\, P'}$$

$$\frac{P \rightarrow_U P'}{Q \rightarrow_U Q'} \quad \text{if } P \equiv Q \quad \text{and} \quad Q' \equiv P'$$

We abbreviate $P \rightarrow_\emptyset P'$ by $P \rightarrow P'$.

The second clause of Definition 4.6 is the most important. In order for an action a to react with a complementary $\bar{a}$, the two sides must not preempt each other (i.e. they must eschew each other's guards). Furthermore the reaction remains conditional on the environment eschewing the union of their guards. The restriction rule shows how this conditionality can then be removed by scoping. Notice that if we restrict attention to the unguarded processes $\mathcal{P}_{\mathsf{Ug}}$ (i.e. we let $U = \emptyset$) we recover the usual CCS reaction relation. So the new transition relation is conservative over the old.

5. Labelled Transitions

As in ordinary CCS, we wish to define a transition relation on processes $P \xrightarrow{\alpha} P'$ meaning that P can perform action α and become P'. However a priority-guarded process can only perform a transition if allowed to by the environment. So, as we did with reaction, we refine the transition relation so that it is parametrised on sets of actions $U \subseteq \mathsf{Pri}$. The intended meaning of $P \xrightarrow{\alpha}_U P'$ is that P can perform α as long as the environment eschews U, i.e. does not offer $\bar{u}$ for any $u \in U$. Our definition is inspired by the transition relation in [CW95], which is parametrised on what set of output actions the environment is ready to perform.

Definition 5.1. (cf [Mil99, Definition 5.1]) The *transition relation* on $\mathcal{P}$ is the smallest relation $\rightarrow$ on $\mathcal{P} \times \mathit{Act} \times \wp(\mathsf{Pri}) \times \mathcal{P}$ generated by the following rules:

$$\text{(sum)} \qquad M + S\!:\!\alpha.P + N \xrightarrow{\alpha}_{S \cap \mathsf{Pri}} P \quad \text{if } \alpha \notin S \cap \mathsf{Pri}$$

$$\text{(react)} \qquad \frac{P_1 \xrightarrow{a}_{U_1} P_1' \quad P_2 \xrightarrow{\bar{a}}_{U_2} P_2' \quad P_1 \,\mathsf{eschews}\, U_2 \quad P_2 \,\mathsf{eschews}\, U_1}{P_1|P_2 \xrightarrow{\tau}_{U_1 \cup U_2} P_1'|P_2'}$$

$$\text{(par)} \qquad \frac{P_1 \xrightarrow{\alpha}_U P_1' \quad P_2 \,\mathsf{eschews}\, U}{P_1|P_2 \xrightarrow{\alpha}_U P_1'|P_2} \qquad \frac{P_2 \xrightarrow{\alpha}_U P_2' \quad P_1 \,\mathsf{eschews}\, U}{P_1|P_2 \xrightarrow{\alpha}_U P_1|P_2'}$$

$$\text{(res)} \qquad \frac{P \xrightarrow{\alpha}_U P'}{\mathrm{new}\ a\, P \xrightarrow{\alpha}_{U-\{a,\bar{a}\}} P'} \quad \text{if } \alpha \notin \{a, \bar{a}\}$$

$$\text{(ident)} \qquad \frac{\{\overrightarrow{b}/\overrightarrow{a}\}P \xrightarrow{\alpha}_U P'}{A\langle\vec{b}\rangle \xrightarrow{\alpha}_U P'} \quad \text{if } A(\overrightarrow{a}) \stackrel{\mathrm{df}}{=} P$$

We abbreviate $P \xrightarrow{\alpha}_{\emptyset} P'$ by $P \xrightarrow{\alpha} P'$ and $\exists P'.P \xrightarrow{\alpha}_U P'$ by $P \xrightarrow{\alpha}_U$.

Proposition 5.2. *If $P \xrightarrow{\alpha}_U P'$ then $\alpha \notin U$ and U is finite. Moreover,*

$$\{u \in \mathsf{Pri} : \exists U.P \xrightarrow{u}_U\} \subseteq \mathsf{off}(P)$$

To see that $\mathsf{off}(P)$ can be unequal to $\{u \in \mathsf{Pri} : \exists U.P \xrightarrow{u}_U\}$, consider $u\!:\!v.0|\bar{u}.0$. We see that $\mathsf{off}(u\!:\!v.0|\bar{u}.0) = \{v, \bar{u}\}$, but $u\!:\!v.0|\bar{u}.0$ cannot perform v.

As with reaction, note that if we restrict attention to the unguarded processes $\mathcal{P}_{\mathsf{Ug}}$ (i.e. we let $U = \emptyset$) we recover the usual CCS transition relation. So the new transition relation is conservative over the old. In applications we envisage that the ordinary CCS transition relation can be used most of the time. The CPG transition relation will only be needed in those subsystems which use priority.

As an illustration of the design choices embodied in our definitions, consider the following example:

$$P \stackrel{\mathrm{df}}{=} u.a.0 + u\!:\!\bar{v}.0 \quad Q \stackrel{\mathrm{df}}{=} v.b.0 + v\!:\!\bar{u}.0 \quad R \stackrel{\mathrm{df}}{=} \mathrm{new}\ u, v\ (P|Q)$$

In P action u has priority over $\bar{v}$, while in Q action v has priority over $\bar{u}$. So the u communication and the v communication have conflicting priorities, with each above the other. This paradoxical example was considered in [CW95] (albeit with a different notation), and the solution adopted there (as we saw in Section 2) was to ban R from being a process. In our approach we have $P \xrightarrow{u} a.0$, $Q \xrightarrow{\bar{u}}_v 0$. For a u communication to happen, by rule (react) we need $\bar{v} \notin \mathsf{off}(P)$, but $\mathsf{off}(P) = \{u, \bar{v}\}$, so that the u communication cannot happen. Similarly the v communication cannot happen, and so $R = 0$.

6. Strong offer bisimulation

Similarly to ordinary CCS, we define process equivalences based on strong and weak bisimulation. We consider strong bisimulation in this section and weak bisimulation (i.e. with hiding of silent actions) in the next.

The intuition behind our notion of bisimulation is that for processes to be equivalent they must make the same offers, and for a process Q to simulate a process P, Q must be able to do whatever P can, though possibly constrained by fewer or smaller priority guards. For instance, we would expect the processes $a.0 + u : a.0$ and $a.0$ to be equivalent, since the priority guarded $u\!:\!a.0$ is simulated by $a.0$.

Definition 6.1. (cf [Mil99]) A relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ is a *strong offer simulation* if $\mathcal{S}(P, Q)$ implies both that $\mathsf{off}(P) = \mathsf{off}(Q)$ and that for all $\alpha \in Act$,

if $P \xrightarrow{\alpha}_U P'$ then for some Q' and $V \subseteq U$, we have $Q \xrightarrow{\alpha}_V Q'$ and $\mathcal{S}(P', Q')$

$\mathcal{S}$ is a strong offer *bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are strong offer simulations.

Definition 6.2. Processes P and Q are *strongly offer equivalent*, written $P \overset{\mathsf{off}}{\sim} Q$, iff there is some strong offer bisimulation $\mathcal{S}$ such that $\mathcal{S}(P, Q)$.

In view of the general theory of bisimulation [Mil99, Section 3.3], $\overset{\mathsf{off}}{\sim}$ is an equivalence relation, and is itself a strong offer bisimulation.

Proposition 6.3. *(cf* [Mil99, Prop 5.2]*)* $\equiv$ *is a simulation. Hence* $\equiv$ *implies* $\overset{\mathsf{off}}{\sim}$*.*

Proof. (Sketch) We must show that if $C[P] \xrightarrow{\alpha}_U P'$ and $P \equiv Q$ is a generating case of Definition 4.1, then $C[Q] \xrightarrow{\alpha}_U Q' \equiv P'$, some Q'. The proof is by cases on the rules for $\rightarrow$ and the generating equations of $\equiv$. The interesting case is $(P|Q)|R \equiv P|(Q|R)$. We consider two example transitions of $(P|Q)|R$, and omit the many other similar cases.

Suppose first that P moves on its own in $(P|Q)|R$, so that

$$(P|Q)|R \xrightarrow{\alpha}_U (P'|Q)|R$$

We have $P \xrightarrow{\alpha}_U P'$ and so $Q \,\mathsf{eschews}\, U$ and $R \,\mathsf{eschews}\, U$. By Proposition 4.5 we deduce that $Q|R \,\mathsf{eschews}\, U$. Hence

$$P|(Q|R) \xrightarrow{\alpha}_U P'|(Q|R)$$

Now suppose that $P \xrightarrow{a}_U P'$ and $Q \xrightarrow{\bar{a}}_V Q'$, and that

$$(P|Q)|R \xrightarrow{\tau}_{U \cup V} (P'|Q')|R$$

We have $P \,\mathsf{eschews}\, V$, $Q \,\mathsf{eschews}\, U$ and $R \,\mathsf{eschews}\, U \cup V$. By Proposition 4.5 we deduce that $R \,\mathsf{eschews}\, V$ and $Q|R \,\mathsf{eschews}\, U$. Therefore $Q|R \xrightarrow{\bar{a}}_V Q'|R$, and

$$P|(Q|R) \xrightarrow{\tau}_{U \cup V} P'|(Q'|R)$$

□

Lemma 6.4. *(cf* [Mil99, Lemma 5.4]*) If* $P \rightarrow_U P'$ *then* $P \xrightarrow{\tau}_U \equiv P'$*.* □

Lemma 6.5. *(cf* [Mil99, Lemma 5.5]*) Let* $P \xrightarrow{a}_U P'$*. Then* P *and* P' *can be expressed, up to* $\equiv$*, as*

$$P \equiv \text{new}\,\vec{z}\,((S\!:\!a.Q + M)|R) \qquad P' \equiv \text{new}\,\vec{z}\,(Q|R)$$

(some $\vec{z}, S, Q, M, R$*) with* $U = S \cap \mathsf{Pri}$*.* □

Theorem 6.6. *(cf* [Mil99, Theorem 5.6]*)*

$$P \xrightarrow{\tau}_U \equiv P' \quad \text{iff} \quad P \rightarrow_U P'$$

Proof. (Sketch) By Lemma 6.4 it is enough to show that $P \xrightarrow{\tau}_U P'$ implies $P \rightarrow_U \equiv P'$. By induction on the proof of $P \xrightarrow{\tau}_U P'$. The interesting case is (react), where we employ Lemma 6.5. ☐

Theorem 6.7. *(cf* [Mil99, Proposition 5.29]*) Strong offer equivalence is a congruence, i.e. if* $P \overset{\mathsf{off}}{\sim} Q$ *then*

1. $S : \alpha.P + M \overset{\mathsf{off}}{\sim} S : \alpha.Q + M$
2. $\text{new}\ a\ P \overset{\mathsf{off}}{\sim} \text{new}\ a\ Q$
3. $P|R \overset{\mathsf{off}}{\sim} Q|R$
4. $R|P \overset{\mathsf{off}}{\sim} R|Q$

Proof. Parallel is the most interesting. Suppose $P \overset{\mathsf{off}}{\sim} Q$. We must show $P|R \overset{\mathsf{off}}{\sim} Q|R$.

Clearly if $\mathsf{off}(Q) = \mathsf{off}(P)$ then $\mathsf{off}(Q|R) = \mathsf{off}(P|R)$.

Suppose $P|R \xrightarrow{\alpha}_U P'|R$ is derived from $P \xrightarrow{\alpha}_U P'$. Then $Q \xrightarrow{\alpha}_{U'} Q'$ with $U' \subseteq U$. So clearly $Q|R \xrightarrow{\alpha}_{U'} Q'|R$.

Now suppose $P|R \xrightarrow{\alpha}_U P|R'$ is derived from $R \xrightarrow{\alpha}_U R'$. Then $P\,\mathsf{eschews}\,U$. So $Q\,\mathsf{eschews}\,U$ and we have $Q|R \xrightarrow{\alpha}_U Q|R'$.

Now suppose $P|R \xrightarrow{\tau}_{U \cup V} P'|R'$ is derived from $P \xrightarrow{a}_U P'$ and $R \xrightarrow{\bar{a}}_V R'$. So $P\,\mathsf{eschews}\,V$ and $R\,\mathsf{eschews}\,U$. Then $Q \xrightarrow{a}_{U'} Q'$ with $U' \subseteq U$. Since $\mathsf{off}(Q) = \mathsf{off}(P)$, $Q\,\mathsf{eschews}\,V$. Also $R\,\mathsf{eschews}\,U'$. Hence $Q|R \xrightarrow{\tau}_{U' \cup V} Q'|R'$.

Summation, restriction, identifier are straightforward. ☐

Note that if $P, Q \in \mathcal{P}_{\mathsf{Ug}}$ then $P \overset{\mathsf{off}}{\sim} Q$ iff $P \sim Q$, where $P \sim Q$ denotes that P and Q are strongly equivalent in the usual sense of [Mil99]. So $\overset{\mathsf{off}}{\sim}$ is conservative over $\sim$. In fact we can say more:

Proposition 6.8. *Let* $P, Q \in \mathcal{P}_{\mathsf{Ug}}$. *If* $P \sim Q$ *then* $C[P] \overset{\mathsf{off}}{\sim} C[Q]$, *for any context* $C[\cdot]$. ☐

So we can reuse all the known equivalences between CCS processes when working with CPG processes.

Proposition 6.9. *(cf* [Mil99, Proposition 5.21]*) For all* $P \in \mathcal{P}$,

$$P \overset{\mathsf{off}}{\sim} \sum\{U : \alpha.Q : P \xrightarrow{\alpha}_U Q\}$$

Proposition 6.10. *(The Expansion Law) For all* $n \geq 0$, *processes* $P_1, \ldots, P_n$ *and names* $\vec{a}$*:*

$$\begin{aligned}
&\text{new}\ \vec{a}\ (P_1 | \cdots | P_n) \overset{\mathsf{off}}{\sim} \\
&\sum\{(U - \{\vec{a}, \bar{\vec{a}}\}) : \alpha.\text{new}\ \vec{a}\ (P_1 | \cdots | P_i' | \cdots | P_n) : 1 \leq i \leq n, P_i \xrightarrow{\alpha}_U P_i', \\
&\bar{U} \cap \mathsf{off}(P_1 | \cdots | P_{i-1} | P_{i+1} | \cdots | P_n) = \{\alpha, \bar{\alpha}\} \cap \vec{a} = \emptyset\} \\
&+ \sum\{((U \cup V) - \{\vec{a}, \bar{\vec{a}}\}) : \tau.\text{new}\ \vec{a}\ (P_1 | \cdots | P_i' | \cdots P_j' | \cdots | P_n) : \\
&1 \leq i < j \leq n, P_i \xrightarrow{b}_U P_i', P_j \xrightarrow{\bar{b}}_V P_j', \\
&\bar{U} \cap \mathsf{off}(P_1 | \cdots | P_{i-1} | P_{i+1} | \cdots | P_n) = \bar{V} \cap \mathsf{off}(P_1 | \cdots | P_{j-1} | P_{j+1} | \cdots | P_n) = \emptyset\}
\end{aligned}$$

Proposition 6.11. *The following laws hold:*

$$M + S : \alpha.P \overset{\mathsf{off}}{\sim} M + (S \cap \mathsf{Pri}) : \alpha.P \tag{6.1}$$

$$M + U : \alpha.P \overset{\mathsf{off}}{\sim} M \qquad \text{if } \alpha \in U \subseteq \mathsf{Pri} \tag{6.2}$$

$$M + U : \alpha.P + (U \cup V) : \alpha.P \overset{\mathsf{off}}{\sim} M + U : \alpha.P \tag{6.3}$$

$$
\begin{array}{rcl}
(\sum U_i : \alpha_i.P_i) & \mid & (\sum V_j : \beta_j.Q_j) \\
(6.4) \qquad & \overset{\mathsf{off}}{\sim} & \sum\{U_i : \alpha_i.(P_i|(\sum V_j : \beta_j.Q_j)) : \forall j.\beta_j \notin \bar{U}_i\} \\
& + & \sum\{V_j : \beta_j.((\sum U_i : \alpha_i.P_i)|Q_j)) : \forall i.\alpha_i \notin \bar{V}_j\} \\
& + & \sum\{(U_i \cup V_j) : \tau.P_i|Q_j : \alpha_i = \bar{\beta}_j \in \mathsf{Vis}, \forall i', j'.\alpha_{i'} \notin \bar{V}_j, \beta_{j'} \notin \bar{U}_i\}
\end{array}
$$

$$
(6.5) \qquad \text{new } a\,(\sum U_i : \alpha_i.P_i) \overset{\mathsf{off}}{\sim} \sum((U_i - \{a, \bar{a}\}) : \alpha_i.\text{new } a\, P_i : \alpha_i \neq a, \bar{a}\}
$$

Definition 6.12. P is in *standard form* if $P \cong \sum U_i : \alpha_i.P_i$ where $\alpha_i \notin U_i$, $U_i \subseteq \mathsf{Pri}$ and each P_i is in standard form. Here $\cong$ means "identically equal". A CPG process is *finite* if it contains no identifiers.

Definition 6.13. Let $\mathcal{A}_S$ be the following set of axioms: the axioms of structural congruence $\equiv$ (Definition 4.1) together with the five laws of Proposition 6.11.

Lemma 6.14. *For any finite process P there is P' in standard form such that $\mathcal{A}_S \vdash P = P'$.*

Lemma 6.15. *If P is in standard form, $P \xrightarrow{\alpha}_U P'$ and $U \subseteq V$ then*

$$\mathcal{A}_S \vdash P = P + V : \alpha.P'$$

Proof. Use law (6.3). $\square$

Theorem 6.16. *The set of axioms $\mathcal{A}_S$ is complete for $\overset{\mathsf{off}}{\sim}$ on finite CPG processes.*

Proof. Given two processes $P \overset{\mathsf{off}}{\sim} Q$ (which can both be taken to be in standard form by Lemma 6.14), we prove them equal by adding to Q a new summand for each summand of P (obtaining $N + Q$), and adding to P a new summand for each summand of Q (obtaining $P + M$).

Suppose $P \cong \sum U_i : \alpha_i.P_i$. For each i, we have $P \xrightarrow{\alpha_i}_{U_i} P_i$. So $Q \xrightarrow{\alpha_i}_{V_i} Q_i'$, some $V_i \subseteq U_i$, $P_i \overset{\mathsf{off}}{\sim} Q_i'$. By Lemma 6.15, $\mathcal{A}_S \vdash Q = Q + U_i : \alpha_i.Q_i'$. By induction on the total depth of P and Q, we have $\mathcal{A}_S \vdash P_i = Q_i'$. Let $N = \sum U_i : \alpha_i.Q_i'$. Then $\mathcal{A}_S \vdash Q = N + Q$. We get M symmetrically, with $\mathcal{A}_S \vdash P = P + M$. Now $\mathcal{A}_S \vdash P + M = N + Q$ and hence $\mathcal{A}_S \vdash P = Q$ as required. $\square$

7. Weak offer bisimulation

We now investigate weak bisimulation, where reactions are hidden.

Definition 7.1. $P \Rightarrow_U P'$ iff $P \cong P'$ or $\exists U_1, \ldots, U_n.P \rightarrow_{U_1} \cdots \rightarrow_{U_n} P'$ with $U = U_1 \cup \cdots \cup U_n$ $(n \geq 1)$.

$P \overset{a}{\Rightarrow}_U P'$ iff $\exists U', U''.P \Rightarrow_{U'} P'' \xrightarrow{a}_{U''} P'$ with $U = U' \cup U''$ and $\mathsf{off}(P'') \subseteq \mathsf{off}(P)$.

Definition 7.2. A relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ is a *weak offer simulation* if $\mathcal{S}(P, Q)$ implies both that $\mathsf{off}(P) = \mathsf{off}(Q)$ and that:

if $P \Rightarrow_U P'$ then for some Q' and $U' \subseteq U$, we have $Q \Rightarrow_{U'} Q'$ and $\mathcal{S}(P', Q')$, and for all $a \in \mathsf{Vis}$,

if $P \overset{a}{\Rightarrow}_U P'$ then for some Q' and $U' \subseteq U$, we have $Q \overset{a}{\Rightarrow}_{U'} Q'$ and $\mathcal{S}(P', Q')$.

$\mathcal{S}$ is a weak offer *bisimulation* if both $\mathcal{S}$ and $\mathcal{S}^{-1}$ are weak offer simulations.

As for CCS, there is a characterisation of weak offer simulation which is more efficient for calculation:

Proposition 7.3. *A relation $\mathcal{S} \subseteq \mathcal{P} \times \mathcal{P}$ is a weak offer simulation iff $\mathcal{S}(P,Q)$ implies both that $\mathsf{off}(P) = \mathsf{off}(Q)$ and that:*

if $P \to_U P'$ then for some Q' and $U' \subseteq U$, we have $Q \Rightarrow_{U'} Q'$ and $\mathcal{S}(P',Q')$, and for all $a \in \mathsf{Vis}$,

if $P \xrightarrow{a}_U P'$ then for some Q' and $U' \subseteq U$, we have $Q \stackrel{a}{\Rightarrow}_{U'} Q'$ and $\mathcal{S}(P',Q')$.

Proof. ($\Rightarrow$) is trivial.

($\Leftarrow$) Suppose $\mathcal{S}$ satisfies the condition of the Proposition. We must show that $\mathcal{S}$ is a weak offer simulation. Clearly $\mathsf{off}(P) = \mathsf{off}(Q)$. It is easy to show

if $P \Rightarrow_U P'$ then for some Q' and $U' \subseteq U$, we have $Q \Rightarrow_{U'} Q'$ and $\mathcal{S}(P',Q')$

by repeated application of the property

if $P \to_U P'$ then for some Q' and $U' \subseteq U$, we have $Q \Rightarrow_{U'} Q'$ and $\mathcal{S}(P',Q')$.

It remains to show

if $P \stackrel{a}{\Rightarrow}_U P'$ then for some Q' and $U' \subseteq U$, we have $Q \stackrel{a}{\Rightarrow}_{U'} Q'$ and $\mathcal{S}(P',Q')$.

Suppose $P \stackrel{a}{\Rightarrow}_U P'$. Then $P \Rightarrow_{U'} P'' \xrightarrow{\alpha}_{U''} P'$ with $U = U' \cup U''$ and $\mathsf{off}(P'') \subseteq \mathsf{off}(P)$. So there are Q'', Q', V', V'' such that $Q \Rightarrow_{V'} Q'' \stackrel{a}{\Rightarrow}_V Q'$ with $V' \subseteq U'$, $V \subseteq U''$, $\mathcal{S}(P'',Q'')$, $\mathcal{S}(P'',Q'')$. We need only establish $Q \stackrel{a}{\Rightarrow}_{V' \cup V} Q'$. But $Q'' \Rightarrow_{V''} Q''' \xrightarrow{a}_{V'''} Q'$ with $\mathsf{off}(Q''') \subseteq \mathsf{off}(Q'')$, $V = V'' \cup V'''$. And $\mathsf{off}(Q'') = \mathsf{off}(P'')$ since $\mathcal{S}(P'',Q'')$. Finally $\mathsf{off}(P'') \subseteq \mathsf{off}(P) = \mathsf{off}(Q)$. So $\mathsf{off}(Q''') \subseteq \mathsf{off}(Q)$. ◻

In view of Proposition 7.3, on the sublanguage $\mathcal{P}_{\mathsf{Ord}}$ (which corresponds to CCS) weak offer simulation is almost the same as for CCS [Mil99, Proposition 6.3]. The difference is that we allow reactions only *before* the visible transition, not both before and after. The definition may also be compared with *branching* and *delay* bisimulation [vGW96].

Definition 7.4. Processes P and Q are *weakly offer equivalent*, written $P \stackrel{\mathsf{off}}{\approx} Q$, iff there is some weak offer bisimulation $\mathcal{S}$ such that $\mathcal{S}(P,Q)$.

Proposition 7.5. *The following hold:*

1. $\stackrel{\mathsf{off}}{\approx}$ is an equivalence relation
2. $\stackrel{\mathsf{off}}{\approx}$ is a weak offer bisimulation
3. For any P, Q, if $P \stackrel{\mathsf{off}}{\sim} Q$ then $P \stackrel{\mathsf{off}}{\approx} Q$.

Proof. 1, 2. Straightforward.

3. One can show that if $\mathcal{S}$ is a strong offer bisimulation then $\equiv \mathcal{S} \equiv$ is a weak offer bisimulation. ◻

Theorem 7.6. *(cf* [Mil99, Proposition 6.17]*) Weak offer equivalence is a congruence.*

Proof. Parallel is the most interesting. Suppose $P \stackrel{\mathsf{off}}{\approx} Q$. We must show $P|R \stackrel{\mathsf{off}}{\approx} Q|R$.

Clearly if $\mathsf{off}(P) = \mathsf{off}(Q)$ then $\mathsf{off}(P|R) = \mathsf{off}(Q|R)$.

Suppose $P|R \xrightarrow{a}_U P'|R$ is derived from $P \xrightarrow{a}_U P'$. Then $Q \Rightarrow_{U'} Q'' \xrightarrow{a}_{U''} Q'$ with $U', U'' \subseteq U$. So clearly $Q|R \Rightarrow_{U'} Q''|R \xrightarrow{a}_{U''} Q'|R$.

Suppose $P|R \to_U P'|R$ is derived from $P \to_U P'$. Then $Q \Rightarrow_{U'} Q'$ with $U' \subseteq U$. So clearly $Q|R \Rightarrow_{U'} Q'|R$.

Now suppose $P|R \xrightarrow{\alpha}_U P|R'$ is derived from $R \xrightarrow{\alpha}_U R'$. Then $\bar{U} \cap \mathsf{off}(P) = \emptyset$. So since $\mathsf{off}(Q) = \mathsf{off}(P)$ we have $Q|R \xrightarrow{\alpha}_U Q|R'$.

Now suppose $P|R \to_{U \cup V} P'|R'$ is derived from $P \xrightarrow{a}_U P'$ and $R \xrightarrow{\bar{a}}_V R'$. Then $Q \Rightarrow_{U'} Q'' \xrightarrow{a}_{U''} Q'$. So $Q|R \Rightarrow_{U'} Q''|R$. Since $\bar{U}'' \subseteq \bar{U}$ and $\mathsf{off}(Q'') \subseteq \mathsf{off}(P)$ we see that $Q''|R \to_{U'' \cup V} Q'|R'$.

Summation, restriction, identifier are straightforward. ◻

So we have a congruence which conservatively extends CCS.

It would have been more obvious to have the following for the clause for $a \in \mathsf{Vis}$ in Definition 7.1:

$P \overset{\alpha}{\Rightarrow}_U P'$ iff there exist P'', U', U'' such that $P \Rightarrow_{U'} P'' \xrightarrow{\alpha}_{U''} P'$, $U = U' \cup U''$ (i.e. omitting the condition $\mathsf{off}(P'') \subseteq \mathsf{off}(P)$).

We would then have defined weak offer bisimulation and weak offer equivalence based on this variant and more generous definition of $P \overset{\alpha}{\Rightarrow}_U P'$. Let $\overset{\mathsf{off}}{\approx}_{\mathsf{var}}$ denote this variant weak offer equivalence.

This would give us a strictly larger equivalence, which would fail to be a congruence. As an example, let

$$P \cong a.0 + \tau.(a.0 + u.0) \qquad Q \cong v : a.0 + \tau.(a.0 + u.0) \qquad R \cong \bar{u} : \bar{a}.b.0 + \bar{v}.0$$

Then $P \overset{\mathsf{off}}{\approx}_{\mathsf{var}} Q$ but not $P \overset{\mathsf{off}}{\approx} Q$. Moreover $P|R \xrightarrow{\tau}_{\bar{u}} 0|b.0$ but $Q|R$ cannot perform a sequence of τs and then b, demonstrating that the variant equivalence is not a congruence.

The congruence induced by the variant version is implied by weak offer equivalence:

Proposition 7.7. *For any P, Q, $P \overset{\mathsf{off}}{\approx} Q$ implies for all contexts $C[\cdot]$,*

$$C[P] \overset{\mathsf{off}}{\approx}_{\mathsf{var}} C[Q]$$

Proof. It is straightforward to show that $P \overset{\mathsf{off}}{\approx} Q$ implies $P \overset{\mathsf{off}}{\approx}_{\mathsf{var}} Q$, and $\overset{\mathsf{off}}{\approx}$ is a congruence. □

We have not determined whether the converse to Proposition 7.7 holds.

We now turn to the equational theory of weak offer equivalence. In CCS we have the law $P \approx \tau.P$ [Mil99, Theorem 6.15]. However in CPG, $u.0 \overset{\mathsf{off}}{\not\approx} \tau.u.0$. This is because $\mathsf{off}(u.0) = \{u\}$ whereas $\mathsf{off}(\tau.u.0) = \emptyset$.

However the usual CCS equivalence laws will still hold for the ordinary processes $\mathcal{P}_{\mathsf{Ord}}$ (recall that for $P \in \mathcal{P}_{\mathsf{Ord}}$, $\mathsf{off}(P) = \emptyset$).

Proposition 7.8. *The following laws hold:*

$$\tau.P \overset{\mathsf{off}}{\approx} P \quad \text{if } \mathsf{off}(P) = \emptyset \tag{7.1}$$

This extends the first τ-law of CCS [Mil99, Theorem 6.15]*: $\tau.P \approx P$.*

$$M + N + \tau.N \overset{\mathsf{off}}{\approx} M + \tau.N \quad \text{if } \mathsf{off}(N) \subseteq \mathsf{off}(M) \tag{7.2}$$

This extends the second τ-law of CCS: $M + N + \tau.N \approx M + \tau.N$.

Note. The third τ-law of CCS:

$$M + \alpha.P + \alpha.(\tau.P + N) \approx M + \alpha.(\tau.P + N)$$

has no analogue in our presentation (since we have a slightly stronger definition of weak bisimulation).

We stated (7.1) and (7.2) because in many situations it is convenient to use conventional CCS reasoning. The next result gives the "intrinsic" τ-laws of CPG:

Proposition 7.9. *The following laws hold:*

$$M + U : \tau.M \overset{\mathsf{off}}{\approx} M \tag{7.3}$$

$$M + U : \tau.(N + V : \tau.P) \overset{\mathsf{off}}{\approx} M + U : \tau.(N + V : \tau.P) + (U \cup V) : \tau.P \tag{7.4}$$

If $\mathsf{off}(N + V\!:\!a.P) \subseteq \mathsf{off}(M)$*:*

$$M + U\!:\!\tau.(N + V\!:\!a.P) \overset{\mathsf{off}}{\approx} M + U\!:\!\tau.(N + V\!:\!a.P) + (U \cup V)\!:\!a.P \tag{7.5}$$

Note. We can derive (7.2) from (7.4) and (7.5). Also we can derive the following form of (7.1):

$$\tau.M \overset{\mathsf{off}}{\approx} M \quad \text{if } \mathsf{off}(M) = \emptyset \tag{7.6}$$

from (7.3), (7.4), (7.5). Recall that every process is strongly equivalent to a summation (Proposition 6.9), and so (7.6) is effectively as strong as (7.1).

Definition 7.10. Let $\mathcal{A}_W$ be the axioms $\mathcal{A}_S$ (Definition 6.13) together with (7.3), (7.4), (7.5).

Lemma 7.11. *(cf* [Mil89, Section 7.4, Lemma 16]*) Let P be in standard form.*

1. If $P \Rightarrow_U P'$ and $U \subseteq V$ then $\mathcal{A}_W \vdash P = P + V\!:\!\tau.P'$
2. If $P \overset{a}{\Rightarrow}_U P'$ and $U \subseteq V$ then $\mathcal{A}_W \vdash P = P + V\!:\!a.P'$

Proof. Use (7.4), (7.5), (6.3). □

Theorem 7.12. *The axioms $\mathcal{A}_W$ are complete for $\overset{\mathsf{off}}{\approx}$ on finite processes.*

Proof. Suppose that $P \overset{\mathsf{off}}{\approx} Q$. By Lemma 6.14 we can assume P and Q are in standard form. We prove P and Q equal much as in Theorem 6.16: For each summand $U\!:\!\alpha.P'$ of P we add a new summand to Q, to form $N + Q$, and for each summand $V\!:\!\alpha.Q'$ of Q we add a new summand to P, to form $P + M$.

Suppose $P \cong \sum U_i\!:\!\alpha_i.P_i$. For each i, there are two cases:

If $\alpha_i = \tau$ we have $P \overset{\tau}{\to}_{U_i} P_i$. So $Q \Rightarrow_{V_i} Q'_i$, some $V_i \subseteq U_i$, $P_i \overset{\mathsf{off}}{\approx} Q'_i$. By Lemma 7.11, $\mathcal{A}_W \vdash Q = Q + U_i\!:\!\tau.Q'_i$. By induction on the total depth of P and Q, we have $\mathcal{A}_W \vdash P_i = Q'_i$ (note that P_i has lower depth than P, even though Q'_i might have the same depth as Q).

If $\alpha_i = a \in \mathsf{Vis}$ we have $P \overset{a}{\to}_U P_i$. Then there is Q'_i such that $P_i \overset{\mathsf{off}}{\approx} Q'_i$ and Q''_i such that $Q \Rightarrow_{U'_i} Q''_i \overset{a}{\to}_{U''_i} Q'_i$ with $U'_i, U''_i \subseteq U_i$ and $\mathsf{off}(Q''_i) \subseteq \mathsf{off}(Q)$. By Lemma 7.11, $\mathcal{A}_W \vdash Q = Q + U_i\!:\!a.Q'_i$. By induction on the total depth of P and Q, we have $\mathcal{A}_W \vdash P_i = Q'_i$.

Let $N = \sum U_i\!:\!\alpha_i.Q'_i$. Then $\mathcal{A}_W \vdash Q = N + Q$. We get M symmetrically, with $\mathcal{A}_W \vdash P = P + M$. Now $\mathcal{A}_W \vdash P + M = N + Q$ and hence $\mathcal{A}_W \vdash P = Q$ as required. □

Proposition 7.13. *(cf* [Mil99, Theorem 6.19]*) Unique solution of equations. Let $\vec{X}$ be a (possibly infinite) sequence of process variables X_i. Up to $\overset{\mathsf{off}}{\approx}$, there is a unique sequence $\vec{P}$ of processes which satisfy the formal equations:*

$$X_i \overset{\mathsf{off}}{\approx} \sum_j U_{ij}\!:\!a_{ij}.X_{k(ij)}$$

(notice that τs are not allowed)

Proof. Much as in [Mil99, Theorem 6.19]. □

8. Example

We now revisit the interrupt example from Section 1. Recall that we had:

$$P \overset{\text{df}}{=} \text{new } \mathsf{mid}, \mathsf{int}_A, \mathsf{int}_B\ (A|B|I) \qquad A \overset{\text{df}}{=} \mathsf{int}_A\!:\!a.\overline{\mathsf{mid}}.A + \mathsf{int}_A.0$$
$$I \overset{\text{df}}{=} \mathsf{int}.(\overline{\mathsf{int}}_A.\overline{\mathsf{int}}_B.0 + \overline{\mathsf{int}}_B.\overline{\mathsf{int}}_A.0) \quad B \overset{\text{df}}{=} \mathsf{int}_B\!:\!b.\mathsf{mid}.B + \mathsf{int}_B.0$$

We want to show $P \overset{\mathsf{off}}{\approx} Q$, where

$$Q \overset{\mathrm{df}}{=} a.Q_1 + b.Q_2 + \mathsf{int}.0 \quad Q_1 \overset{\mathrm{df}}{=} b.Q + \mathsf{int}.0 \quad Q_2 \overset{\mathrm{df}}{=} a.Q + \mathsf{int}.0$$

Clearly $\mathsf{int}_A, \mathsf{int}_B \in \mathsf{Pri}$. We take $a, b, \mathsf{mid}, \mathsf{int} \in \mathsf{Ord}$. This means that $Q \in \mathcal{P}_{\mathsf{Ord}}$. We can use the Expansion Law (Proposition 6.10) to get:

$$P \overset{\mathsf{off}}{\sim} a.P_1 + b.P_2 + \mathsf{int}.P_3$$

$$P_1 \overset{\mathsf{off}}{\sim} b.P_4 + \mathsf{int}.\tau.0 \quad P_2 \overset{\mathsf{off}}{\sim} a.P_4 + \mathsf{int}.\tau.0 \quad P_3 \overset{\mathsf{off}}{\sim} \tau.\tau.0 + \tau.\tau.0 \quad P_4 \overset{\mathsf{off}}{\sim} \tau.P + \mathsf{int}.\tau.P_3$$

where P_1, P_2, P_3, P_4 are various states of P. We can use law (7.1) together with law (6.3) to get:

$$P \overset{\mathsf{off}}{\sim} a.P_1 + b.P_2 + \mathsf{int}.P_3$$

$$P_1 \overset{\mathsf{off}}{\sim} b.P_4 + \mathsf{int}.0 \quad P_2 \overset{\mathsf{off}}{\sim} a.P_4 + \mathsf{int}.0 \quad P_3 \overset{\mathsf{off}}{\sim} 0 \quad P_4 \overset{\mathsf{off}}{\sim} \tau.P + \mathsf{int}.0$$

Then we use law (7.2) to get $\tau.P + \mathsf{int}.0 \overset{\mathsf{off}}{\approx} \tau.P$. Notice that this needs $\mathsf{off}(P) = \emptyset$, i.e. $a, b, \mathsf{int} \notin \mathsf{Pri}$. Finally:

$$P \overset{\mathsf{off}}{\approx} a.P_1 + b.P_2 + \mathsf{int}.0 \quad P_1 \overset{\mathsf{off}}{\approx} b.P + \mathsf{int}.0 \quad P_2 \overset{\mathsf{off}}{\approx} a.P + \mathsf{int}.0$$

By Proposition 7.13 we get $P \overset{\mathsf{off}}{\approx} Q$ as we wanted.

Our reasoning was presented equationally, but could equally well have been done using bisimulation. We first used the Expansion Law for CPG to unfold the behaviour of P. Since all prioritised actions were restricted, the system P had no priorities as far as the environment was concerned. We could therefore remove silent actions and simplify using standard techniques of CCS.

Note. In Sections 1 and 2 we used plain equality when talking about equivalence between CPG processes, for example, $\mathsf{Sys} = P$. This is to be interpreted as $\overset{\mathsf{off}}{\approx}$.

9. Logic

We briefly consider Hennessy-Milner-style logics for specifying properties of CPG processes. Such logics have previously been defined for a distributed prioritised process algebra [CLN98, Lüt98].

Our starting point is the logic for CCS defined by Hennessy and Milner [HM85, Mil89]. This allows us to describe properties which processes may enjoy. For instance $\langle a\rangle\mathsf{T}$ would mean "can do action a". In our setting, actions are conditional on the environment eschewing certain actions, and so it is natural to generalise $\langle a\rangle\mathsf{T}$ to $\langle a\rangle_U\mathsf{T}$, meaning "can do action a provided the environment eschews the set U". We further augment Hennessy-Milner Logic (HML) with propositions expressing eschewing—$\mathsf{eschew}(U)$ means "eschews U".

Definition 9.1. Strong Prioritised HML. $\mathcal{L}_S$ is the smallest class of formulas containing the following, where $U \subseteq \mathsf{Pri}$ is a finite set and it is assumed that φ and ψ are already in $\mathcal{L}_S$:

1. $\mathsf{eschew}(U)$ (eschewing)
2. $\langle\alpha\rangle_U\varphi$, $\alpha \in \mathsf{Act}$ (possibility)
3. $\neg\varphi$ (negation)
4. $\varphi \wedge \psi$ (conjunction)

Definition 9.2. The *satisfaction* relation $\models \subseteq \mathcal{P} \times \mathcal{L}_S$ is defined as follows:

1. $P \models \mathsf{eschew}(U)$ iff $P\,\mathsf{eschews}\,U$
2. $P \models \langle\alpha\rangle_U\varphi$ iff $\exists P', V \subseteq U. P \xrightarrow{\alpha}_V P' \models \varphi$

3. $P \models \neg\varphi$ iff $P \not\models \varphi$
4. $P \models \varphi \wedge \psi$ iff $P \models \varphi$ and $P \models \psi$

We let T (true) abbreviate the empty conjunction, which is satisfied by every process. Incidentally, $\mathsf{eschew}(\emptyset)$ is also satisfied by every process.

As with CCS [Mil89, Section 10.5], strong offer equivalence is characterised by $\mathcal{L}_S$, in the following sense:

Proposition 9.3. *Let $P, Q \in \mathcal{P}$. Then $P \overset{\mathsf{off}}{\sim} Q$ iff*

$$\forall \varphi \in \mathcal{L}_S.(P \models \varphi \Longleftrightarrow Q \models \varphi)$$

Proof. We show that $\mathsf{off}(P) = \mathsf{off}(Q)$ iff for all finite $U \subseteq \mathsf{Pri}$, $P \models \mathsf{eschew}(U) \Longleftrightarrow Q \models \mathsf{eschew}(U)$. The rest of the proof is much as in CCS, and we omit the details. $\square$

We can also characterise weak offer equivalence using Weak Prioritised HML $\mathcal{L}_W$. This is defined as in Definition 9.1, except that, in Clause 2, $\langle a \rangle_U \varphi$ is replaced by $\langle\langle a \rangle\rangle_U \varphi$ ($a \in \mathsf{Vis}$), and $\langle \tau \rangle_U \varphi$ is replaced by $\langle\langle\rangle\rangle_U \varphi$. We also allow infinitary conjunction in Clause 4—this is because, unlike in the strong case, the transition system in the weak case is not *image-finite*. That is, for a given action a and process P, the set $\{P' : \exists U.P \overset{a}{\Rightarrow}_U P'\}$ may be infinite.

Definition 9.4. Weak Prioritised HML. $\mathcal{L}_W$ is the smallest class of formulas containing the following, where $U \subseteq \mathsf{Pri}$ is a finite set and it is assumed that φ and φ_i are already in $\mathcal{L}_W$:

1. $\mathsf{eschew}(U)$ (eschewing)
2. $\langle\langle a \rangle\rangle_U \varphi$, $a \in \mathsf{Vis}$ (possibility)
   $\langle\langle\rangle\rangle_U \varphi$
3. $\neg\varphi$ (negation)
4. $\bigwedge_{i \in I} \varphi_i$, I a possibly infinite index set (conjunction)

The satisfaction relation is the obvious modification of Definition 9.2:

Definition 9.5. The *satisfaction* relation $\models \subseteq \mathcal{P} \times \mathcal{L}_W$ is defined as follows:

1. $P \models \mathsf{eschew}(U)$ iff $P\, \mathsf{eschews}\, U$
2. $P \models \langle\langle\rangle\rangle_U \varphi$ iff $\exists P', V \subseteq U.P \Rightarrow_V P' \models \varphi$
   $P \models \langle\langle a \rangle\rangle_U \varphi$ iff $\exists P', V \subseteq U.P \overset{a}{\Rightarrow}_V P' \models \varphi$
3. $P \models \neg\varphi$ iff $P \not\models \varphi$
4. $P \models \bigwedge_{i \in I} \varphi_i$ iff $\forall i \in I.P \models \varphi_i$

Clearly the logics $\mathcal{L}_S$ and $\mathcal{L}_W$ share some formulas, but in these cases the respective satisfaction relations agree with each other.

Proposition 9.6. *Let $P, Q \in \mathcal{P}$. Then $P \overset{\mathsf{off}}{\approx} Q$ iff*

$$\forall \varphi \in \mathcal{L}_W.(P \models \varphi \Longleftrightarrow Q \models \varphi)$$

Proof. Much as for Proposition 9.3. $\square$

10. Conclusions

We have introduced priority guards into the summation operator of CCS to form the language CPG. We have defined both strong and weak bisimulation equivalences and seen that they are conservative over the CCS equivalences, and that they are congruences. We have given complete equational laws for finite CPG in both the strong and weak cases. Conservation over CCS has the consequence that in verifying CPG systems we can often use normal CCS reasoning, as long as we take some care with actions in the set of prioritised actions Pri. We have formulated Hennessy-Milner-style logics and seen that they characterise our bisimulation equivalences.

We have seen that CPG overcomes the asymmetry between inputs and outputs present both in Camilleri and Winskel's calculus and in the corresponding calculus of Cleaveland, Lüttgen and Natarajan. Also preemption in CPG is handled rather differently from the calculi just mentioned, in that it depends entirely on the environment.

Acknowledgement. We wish to thank Rajagopal Nagarajan and Philippa Gardner for helpful discussions and suggestions.

References

[BBK86] J.C.M. Baeten, J. Bergstra, and J.-W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra. *Fundamenta Informaticae*, 9:127–168, 1986.

[BK84] J. Bergstra and J.-W. Klop. Process algebra for synchronous communication. *Information and Computation*, 60:109–137, 1984.

[CH90] R. Cleaveland and M.C.B. Hennessy. Priorities in process algebra. *Information and Computation*, 87(1/2):58–77, 1990.

[CLN98] R. Cleaveland, G. Lüttgen, and V. Natarajan. A process algebra with distributed priorities. *Theoretical Computer Science*, 195(2):227–258, 1998.

[CLN00] R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebra. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*. Elsevier, 2000. To appear.

[CW95] J. Camilleri and G. Winskel. CCS with priority choice. *Information and Computation*, 116(1):26–37, 1995.

[Fid93] C.J. Fidge. A formal definition of priority in CSP. *ACM Transactions on Programming Languages and Systems*, 15(4):681–705, 1993.

[HM85] M.C.B. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, 32(1):137–161, 1985.

[HO92] H. Hansson and F. Orava. A process calculus with incomparable priorities. In *Proceedings of the North American Process Algebra Workshop*, pages 43–64, Stony Brook, New York, 1992. Springer-Verlag Workshops in Computer Science.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[Jef93] A. Jeffrey. A typed, prioritized process algebra. Technical Report 13/93, Dept. of Computer Science, University of Sussex, 1993.

[Lüt98] G. Lüttgen. *Pre-emptive Modeling of Concurrent and Distributed Systems*. PhD thesis, Universität Passau, Passau, Germany, May 1998.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[Mil99] R. Milner. *Communicating and Mobile Systems: the π-calculus*. Cambridge University Press, 1999.

[NCCC94] V. Natarajan, L. Christoff, I. Christoff, and R. Cleaveland. Priorities and abstraction in process algebra. *Lecture Notes in Computer Science*, 880:217–230, 1994.

[Pra94] K.V.S. Prasad. Broadcasting with priority, 5th ESOP. *Lecture Notes in Computer Science*, 788:469–484, 1994.

[SS96] S. Smolka and B. Steffen. Priority as extremal probability. *Formal Aspects of Computing*, 8:585–606, 1996.

[vGW96] Rob J. van Glabbeek and W. Peter Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the Association for Computing Machinery*, 43(3):555–600, May 1996.