Reversible Barbed Congruence on Configuration Structures *

Clément Aubert
INRIA
Université Paris-Est, LACL (EA 4219), UPEC, F-94010 Créteil, France
clement.aubert@lacl.fr

Ioana Cristescu
Univ. Paris Diderot, Sorbonne Paris Cité, P.P.S., UMR 7126, F-75205 Paris, France
ioana.cristescu@pps.univ-paris-diderot.fr

A standard contextual equivalence for process algebras is strong barbed congruence. Configuration structures are a denotational semantics for processes in which one can define equivalences that are more discriminating, i.e. that distinguish the denotation of terms equated by barbed congruence. Hereditary history preserving bisimulation (HHPB) is such a relation. We define a strong back and forth barbed congruence using a reversible process algebra and show that the relation induced by the back and forth congruence is equivalent to HHPB, providing a contextual characterization of HHPB.

Introduction

A standard notion of equivalence for process algebras identifies processes that interact the same way with the environment. Reduction congruence [7] is a standard relation that equates terms capable of simulating each other's reductions in any context. However observing only the reductions is a too coarse relation. A predicate, called a *barb*, is then defined to handle an extra observation on processes: the channel on which they communicate with the environment.

Configuration structures—also called *stable families* [12] or *stable configuration structures* [4]—are an extensional representation of processes, which explicit all possible future behaviours. It consists of a family of sets, where each set is called a *configuration* and stands for a reachable state in the run of the process. The elements of the sets, called *events*, represent the actions the process triggered so far. The inclusion relation between configurations stands for the possible paths followed by the execution. The encoding of terms of the Calculus of Communicating Systems (CCS)—a simple process algebra—in configuration structures [12, 4] settled configuration structures as a denotational model for concurrency.

Configuration structures are "true concurrency" models, as opposed to process algebras, which use an interleaving representation of concurrency. It is hard to deduce in an interleaving semantics the relationships between events, such as whether two events are independent or not, whereas they are explicit or easily inferred in a truly concurrent semantics.

On such structures, the equivalence relations defined are more discriminating: it is possible to move "up and down" in the lattice, whereas in the operational semantics, only forward transitions have to be simulated. As an example, consider the processes $a.0|b.0$ and $a.b.0 + b.a.0$ that are bisimilar in CCS but whose causal relations between events differ. In particular we investigate hereditary history preserving bisimulation (HHPB), which equates structures that can simulate each others' forward and backward moves. It is the canonical equivalence on configuration structures as it respects the causality and concurrency relations between events and admits a categorical representation [5].

*This work was partly supported by the ANR-14-CE25-0005 ELICA and the ANR-11-INSE-0007 REVER.

S. Knight, I. Lanese, A. Lluch Lafuente and H. T. Vieira (Eds.):
8th Interaction and Concurrency Experience (ICE 2015)
EPTCS 189, 2015, pp. 68–85, doi:10.4204/EPTCS.189.7

Reversibility allows to define HHPB in an operational setting, by simply adding to processes the capability to undo previous computations. A term can then either continue its forward execution or backtrack up to a point in the past and resume from there. Reversible process algebras are interesting in their own right [6, 1], but we focus in this paper on their capability to simulate the back-and-forth behaviour of configuration structures. To ensure that the backward reduction of CCS indeed corresponds to the backward moves of its denotational representation, one has to prove that the *labelled transition system* is *prime* [8]. It was already done for CCSK [9], a reversible calculus that has a causal consistent backtracking machinery. In this paper we use RCCS [2] a causal consistent reversible variant of CCS whose syntax is given in Sect. 1.1.

In reversible calculi one is also interested in a contextual equivalence for processes. Traditional equivalences, defined only on forward transitions, are inappropriate for processes that can do back-and-forth reductions. Strong back-and-forth bisimulation [6] is more adapted but it is not contextual. Hence we introduce the barbed back-and-forth congruence on RCCS terms (Sect. 1.2) which corresponds to the barbed congruence of CCS except that backwards reductions are also observed.

Configuration structures (Sect. 2) lacks a notion of contextual equivalence, because the context is a notion specific to the operational semantics. Hence it makes sense to consider context only for configuration structures that represent an operational term (Sect. 3). We introduce in Sect. 4 the correct notions and relations on those structures. The contextual equivalence on processes induces a relation on the denotation of these processes and this relation corresponds to HHPB (Sect. 5).

Similarly to the proof in CCS, the correspondence between a contextual equivalence and a non contextual one necessits to approximate hhbp with (a family of) inductive relations defined on configuration structures. If we are interested only in the forward direction (as in CCS), the inductive reasoning starts with the empty set, and constructs the bisimilarity relation by adding pairs of configuration reachable in the same manner from the empty set. However, to approximate hhbp, we need to have an inductive reasoning on the backward transition as well (Definition 15). These relations are of major importance to prove our main theorem (Theorem 1), as they re-introduce the possibility of an inductive reasoning thanks to a stratification of the HHPB relation.

Hhpb is equivalent to strong bisimulation on reversible CCS [10], thus it can be characterised as a non contextual equivalence on processes. One can then prove the main result of the paper by showing that in RCCS strong bisimulation and strong barbed congruence equate the same terms. We chose to use configuration structures instead, as we plan to investigate weak equivalences on reversible process algebra and their correspondence in denotational semantics.

Our work is restrained to processes that forbid *some sort of auto-concurrency* (see Remark 1) and that are "collapsed" (Definition 10): we need to uniquely identify open configurations using only the label and the order of the events. The "equidepth auto-concurrency" [10] does not help.

1 RCCS syntax and bisimulation

RCCS is a reversible variant of CCS, that allows computations to backtrack, hence introducing the notions of *forward* and *backward* transitions. A mechanism of memories attached to processes store the relevant information to eventually do backward steps.

In a sequential setting backtracking follows the exact order of the forward computation. This is too strict for a concurrent calculus where independent processes can fire independent actions. The order of these actions in the forward direction is just *temporal* and *not causal*, and thus it should be allowed to backtrack them in any order. On the other hand, too much liberty in backtracking could allow the system

$$\gamma := a \parallel \bar{a} \parallel \ldots \qquad \alpha,\beta := \gamma \parallel \tau \qquad \text{(Actions)}$$
$$m := \emptyset \parallel \curlyvee.m \parallel \langle i,\alpha,P\rangle.m \qquad \text{(Memories)}$$
$$P,Q := 0 \parallel \alpha.P \parallel \alpha.P+\beta.Q \parallel P|Q \parallel (a)P \qquad \text{(CCS processes)}$$
$$R,S := m \rhd P \parallel R|R \parallel (a)R \qquad \text{(RCCS processes)}$$

Figure 1: RCCS processes grammar

to access states that were not reachable with forward transitions alone.

1.1 RCCS syntax

Notations 1. Let $\mathsf{N} = \{a,b,\ldots\}$ a set of *names*, $\mathsf{I} = \{i,j,\ldots\}$ a set of *identifiers*. An *action* is an input (resp. output) on a channel a, labelled a (resp. $\bar{a}$), or a synchronisation with the label $(a,\bar{a})$, sometimes denoted τ. Each action a has a dual written $\bar{a}$, we let $\bar{\bar{a}} = a$ and $\bar{\tau} = \tau$. Denote $\mathsf{L} = \{\alpha,\beta,..\}$ the set of *labels*.

CCS processes are build using prefix, sum, parallel composition and restriction. RCCS processes, also called *monitored processes*, are built upon CCS processes by adding a memory m that acts as a stack of the previous computations. Each entry in the memory is called an *event* and has a unique identifier. The usual [3] RCCS processes grammar is recalled in Figure 1. A memory $\langle i,\alpha,P\rangle$ contains an "identifier" i that "tags" transitions: it is especially useful in the case of synchronisation (both forward and backward), for it identifies which two processes interact. The label α marks which action has been fired (in the case of a forward transition), or what action should be restored (in the case of a backward move). Finally, P saves the whole process that has been erased when firing a sum. The "fork symbol" $\curlyvee$ marks that the memory of a parallel composition has been split down to the two parts of the parallel composition. It was handled with $\langle 1\rangle$ and $\langle 2\rangle$ (Left- and Right-fork) in previous work [2, p. 295].

We can easily retrieve a CCS process from an RCCS one by erasing the memories:

$$\varepsilon(m \rhd P) = P \quad \varepsilon(R|S) = \varepsilon(R)|\varepsilon(S) \quad \varepsilon((a)R) = (a)\varepsilon(R) \quad \varepsilon(R+S) = \varepsilon(R)+\varepsilon(S)$$

Structural congruence on monitored processes is the smallest equivalence relation up to uniform renaming of identifiers generated by the following rules:

$$\frac{P \equiv Q}{\emptyset \rhd P \equiv \emptyset \rhd Q} \qquad \begin{array}{l} m \rhd (P|Q) \equiv (\curlyvee.m \rhd P | \curlyvee.m \rhd Q) \\ m \rhd (a)P \equiv (a)m \rhd P \text{ with } a \notin m \end{array}$$

The left rule implies that all equivalence for CCS processes holds for RCCS processes with an empty memory. The right rules respectively distributes the memory between two forking processes (top) and moves the restrictions at the process level (bottom).

The labelled transition system (LTS) for RCCS is given by the rules of Figure 2. In the transitions $\xrightarrow{i:\alpha}$ (resp. $\overset{i:\alpha}{\rightsquigarrow}$) for the *forward* (resp. *backward*) action, we have that $i \in I$ is the event identifier, $\mathsf{I}(m)$ (resp. $\mathsf{I}(S)$) is the set of identifiers occurring in m (resp. in S). We use $\overset{i:\alpha}{\twoheadrightarrow}$ as a wildcard for $\xrightarrow{i:\alpha}$ or $\overset{i:\alpha}{\rightsquigarrow}$, and if there are indices $i_1,\ldots,i_n$ and labels $\alpha_1,\ldots,\alpha_n$ such that $R_1 \overset{i_1:\alpha_1}{\twoheadrightarrow} \ldots \overset{i_n:\alpha_n}{\twoheadrightarrow} R_n$, then we write $R_1 \twoheadrightarrow^\star R_n$. We sometimes omit the identifier or the label in the transition. The trace is unique up to renaming of the indices.

$$\frac{R \overset{i:\gamma}{\twoheadrightarrow} R' \qquad S \overset{i:\bar{\gamma}}{\twoheadrightarrow} S'}{R|S \overset{i:\tau}{\twoheadrightarrow} R'|S'}\ \text{syn.} \qquad \frac{R \overset{i:\alpha}{\twoheadrightarrow} R'}{R|S \overset{i:\alpha}{\twoheadrightarrow} R'|S}\ \text{par.} \qquad \frac{R \overset{i:\alpha}{\twoheadrightarrow} R' \qquad a \notin \alpha}{(a)R \overset{i:\alpha}{\twoheadrightarrow} (a)R'}\ \text{res.} \qquad \frac{R_1 \equiv R \overset{i:\alpha}{\twoheadrightarrow} R' \equiv R'_1}{R_1 \overset{i:\alpha}{\twoheadrightarrow} R'_1} \equiv$$

$$\frac{}{m \rhd \alpha.P + Q \overset{i:\alpha}{\longrightarrow} \langle i, \alpha, Q\rangle.m \rhd P}\ \text{act.} \qquad \frac{}{\langle i, \alpha, Q\rangle.m \rhd P \overset{i:\alpha}{\rightsquigarrow} m \rhd \alpha.P + Q}\ \text{act.}_*$$

The rule act. and act$_*$ apply iff $i \notin \mathsf{I}(m)$, the rule par. applies iff $i \notin \mathsf{I}(S)$.

Figure 2: Rules of the LTS

When a prefix is consumed we add in the memory an event consisting of an unique identifier, the label consumed and the discarded part of the non-deterministic sum. Then backtracking removes an event at the top of a memory and restores the prefix and the non-deterministic sum. Synchronization, forward or backward (syn), requires the two synchronization partners to agree on the event identifier and trigger the transitions simultaneously. The requirement that $i \notin \mathsf{I}(S)$ for the parallel composition (par.) ensures the uniqueness of the event identifiers in the forward direction and prevents a part of a previous synchronization to backtrack alone in the backward direction.

Example 1.1. The process $\curlyvee.\langle i, \alpha, \alpha'.0\rangle.\emptyset \rhd P \mid \langle j, \beta, \beta'.0\rangle.\emptyset \rhd Q$ highlights that not all syntactically correct processes have an operational meaning. This term cannot be obtained by a forward computation from a CCS process, somehow "its memory is broken". Without $\curlyvee$, one could backtrack to $\emptyset \rhd \alpha.P + \alpha'.0 \mid \emptyset \rhd \beta.Q + \beta'.0$, but this terms violate the structural congruence.

The semantically correct processes are called *coherent* and are defined as follows:

Definition 1 (Coherent process and O_R)**.** A RCCS process R is *coherent* if there exists a CCS process P such that $\emptyset \rhd P \longrightarrow^\star R$. This process P is unique up to structural congruence and we write it O_R.

Backtracking is not deterministic, but it is noetherian and confluent [3, Lemma 1], hence the uniqueness. Actually, coherence of processes comes from the coherence relation defined on memories [2, Definition 1] and implies that in a coherent term, memories are unique. Moreover, coherence is preserved by transitions and structural congruence.

1.2 A contextual equivalence for RCCS

Let us now revisit the barbed congruence of CCS [7] in the case of RCCS. For that we need the right notions of context and barb in the reversible setting.

Choosing the right notion of context is subtle. A context has to become an executable process regardless of the process instantiated with it. We can distinguish three types of contexts: with an empty memory, with a non empty but coherent memory (i.e. the context can backtrack up to an empty memory regardless of the process instantiated with) or with a non coherent memory. The later is left as future work, while the first two are equivalent: we will only, w.l.o.g., consider contexts without memory.

Definition 2 (CCS Context)**.** A context is a process with a hole: $C := [\,] \parallel \alpha.C \parallel C + P \parallel C|P \parallel (a)C$

We can only instantiate a context with an RCCS process R if the process has an empty memory, i.e. $R = \emptyset \rhd P$. We use the notation $C[\emptyset \rhd P]$ to denote the process $\emptyset \rhd C[P]$.

Definition 3 (Strong commitment (barb))**.** We write $R \downarrow_\alpha$ if there exists $i \in I$ and R' such that $R \overset{i:\alpha}{\longrightarrow} R'$.

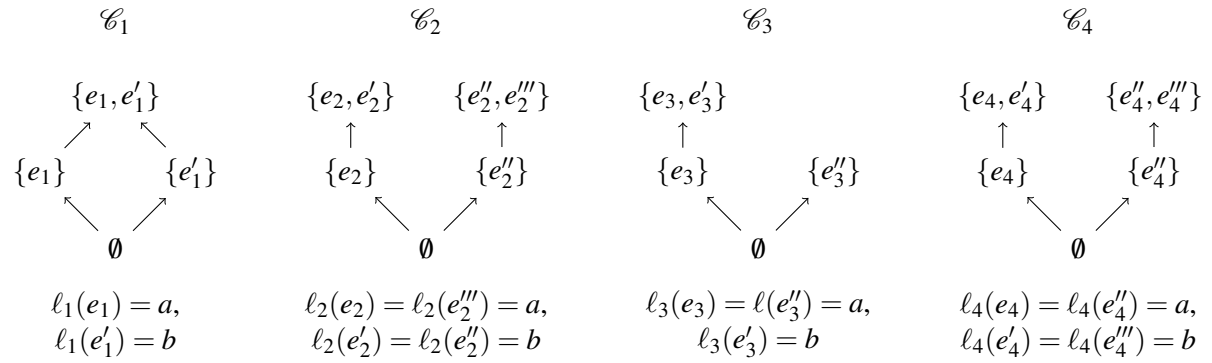


Figure 3: Four examples of configuration strutures

Definition 4. A *strong back-and-forth barbed bisimulation* is a symmetric relation on coherent processes $\dot{\sim}^{\tau}$ such that if $R \dot{\sim}^{\tau} S$, then

$$R \overset{i:\tau}{\rightsquigarrow} R' \implies \exists S' \text{ s.t. } S \overset{i:\tau}{\rightsquigarrow} S' \text{ and } R' \dot{\sim}^{\tau} S' \qquad \text{(Back)}$$

$$R \xrightarrow{i:\tau} R' \implies \exists S' \text{ s.t. } S \xrightarrow{i:\tau} S' \text{ and } R' \dot{\sim}^{\tau} S' \qquad \text{(Forth)}$$

$$R \downarrow_a \implies S \downarrow_a . \qquad \text{(Barbed)}$$

We write $R \sim^{\tau} S$ and define the *strong back-and-forth barbed congruence* if $R \dot{\sim}^{\tau} S$ and for all context $C[\cdot], C[O_R] \dot{\sim}^{\tau} C[O_S]$.

Lemma 1. $R \sim^{\tau} S \implies O_R \sim^{\tau} O_S$.

The proof is straightforward. The converse does not hold as R and S can be any derivative of O_R.

2 Configuration structures

We use configuration structures [12, 4] as a denotational semantics for processes. We recall the definitions and the operations necessary to encode processes, and refer to Winskel's work for the proofs.

Notations 2. Let E be a set, $\subseteq$ be the usual set inclusion relation and C be a family of subsets of E. For $X \subseteq C$ we say that X is *compatible* and write $X \uparrow^{\text{fin.}}$ if $\exists y \in C$ finite such that $\forall x \in X, x \subseteq y$.

Definition 5. A *configuration structure* $\langle E, C \rangle$ is a set E and $C \subseteq \mathscr{P}(E)$ satisfying:

$$\forall x \in C, \forall e \in x, \exists z \in C \text{ finite s.t. } e \in z \text{ and } z \subseteq x \qquad \text{(finitness)}$$

$$\forall x \in C, \forall e, e' \in x, \text{ if } e \neq e' \text{ then } \exists z \in C, z \subseteq x \text{ and } (e \in z \iff e' \notin z) \qquad \text{(coincidence freeness)}$$

$$\forall X \subseteq C \text{ and } X \uparrow^{\text{fin.}} \Rightarrow \cup X \in C \qquad \text{(finite completness)}$$

$$\forall x, y \in C, \text{ if } x \cup y \in C \text{ then } x \cap y \in C \qquad \text{(stability)}$$

A *labelled configuration structure* $\mathscr{C} = \langle E, C, \ell \rangle$ is a configuration structure endowed with a *labelling function* $\ell : E \to \mathsf{L}$. All configurations structures from now on will be supposed to be labelled.

The elements of E are called *events* and subsets of C *configurations*. Intuitively, events are the actions occurring during the run of a process, while a configuration represents a state reached at some point.

Example 2.1. In Figure 3, the configuration structure $\mathscr{C}_1$ have two events e_1, e'_1, with labels respectively a and b, that are concurrent. Configuration $\{e_1\}$ then corresponds to the process that fired action a. Its only possibility is then to fire b and reach the state $\{e_1, e'_1\}$. A process corresponding to this structure is $a.0|b.0$. The configuration structure $\mathscr{C}_2$ corresponds to a process where the events labelled respectively a and b are causally dependent, as in $a.b.0 + b.a.0$.

The configuration structure corresponding to a process P is defined inductively on the syntax of P. Hence the encoding of a process is built from the encoding of its parts, unlike other models such as process graphs (or prime graphs) for CSSK [9]. Moreover, configuration structures are *compositional* in the sense that we can compose configuration structures into new structures. Compositionality is an important feature as it allows us to reason on the context of a process.

Henceforth we detail how the operations of process algebras are translated on configuration structures, which in some cases have a nice categorical interpretation. While the underlying category theory is not used in the paper, it can help in understanding how these structures behave.

Definition 6 (Category of labelled configuration structures)**.** A morphism of labelled configurations structures $f : \langle E_1, C_1, \ell_1\rangle \to \langle E_2, C_2, \ell_2\rangle$ is a partial function on the underlying sets $f : E_1 \to E_2$ that is:

$$\forall x \in C_1, f(x) = \{f(e) \mid e \in x\} \in C_2 \qquad \text{(configuration preserving)}$$
$$\forall x \in C_1, \forall e_1, e_2 \in x, f(e_1) = f(e_2) \implies e_1 = e_2 \qquad \text{(locally injective)}$$
$$\forall x \in C_1, \forall e \in x, \ell_1(e) = \ell_2(f(e)) \qquad \text{(label preserving)}$$

The configuration structures and their morphisms form a category.

Definition 7 (Operation on configuration structures [12])**.** We let $\mathscr{C}_1 = \langle E_1, C_1, \ell_1\rangle$, $\mathscr{C}_2 = \langle E_2, C_2, \ell_2\rangle$ be two configuration structures, set $E^\star = E \cup \{\star\}$ and define the following operations:

Product Define *the product of $\mathscr{C}_1$ and $\mathscr{C}_2$* as $\mathscr{C} = \mathscr{C}_1 \times \mathscr{C}_2$, for $\mathscr{C} = \langle E, C, \ell\rangle$, where $E = E_1^\star \times E_2^\star$ is the product on sets with the projections π_1, π_2 and

$$x \in C \iff \begin{cases} \pi_1(x) \in C_1 \text{ and } \pi_2(x) \in C_2, \\ \pi_1 : \mathscr{C} \to \mathscr{C}_1 \text{ and } \pi_2 : \mathscr{C} \to \mathscr{C}_2 \text{ are morphisms,} \\ x \text{ satisfies (finitness) and (coincidence freeness).} \end{cases}$$

The labelling function ℓ is defined as $\ell(e) = (\ell_1(e_1), \ell_2(e_2))$, where $\pi_1(e) = e_1$ and $\pi_2(e) = e_2$.

Coproduct Define *the coproduct of $\mathscr{C}_1$ and $\mathscr{C}_2$* as $\mathscr{C} = \mathscr{C}_1 + \mathscr{C}_2$, for $\mathscr{C} = \langle E, C, \ell\rangle$, where $E = (\{1\} \times E_1) \cup (\{2\} \times E_2)$ and $C = \{\{1\} \times x \mid x \in C_1\} \cup \{\{2\} \times x \mid x \in C_2\}$. The labelling function ℓ is defined as $\ell(e) = \ell_i(e_i)$ when $e_i \in E_i$ and $\pi_i(e_i) = e$.

Restriction Let $E' \subseteq E$. Define *the restriction of a set of events* as $\langle E, C, \ell\rangle \restriction E' = \langle E', C', \ell'\rangle$ where $x' \in C' \iff x \in C, x \subseteq E'$. *The restriction of a name* is then $\langle E, C, \ell\rangle \restriction E_a$ where $E_a = \{e \in E \mid \ell(e) \neq \tau, a \in \ell(e)\}$.

Prefix Define *the prefix operation on configuration structures* as $\alpha.\langle E, C, \ell\rangle = \langle e \cup E, C', \ell'\rangle$, for $e \notin E$ where $x' \in C' \iff \exists x \in C, x' = x \cup e$ and $\ell'(e) = \alpha$, and $\forall e' \neq e$, $\ell'(e') = \ell(e')$.

Relabelling Define *the relabelling of a configuration structure* as $\mathscr{C}_1 \circ \ell = \langle E_1, C_1, \ell_1 \circ \ell \rangle$, where ℓ is a labelling function.

Parallel composition Define $\mathscr{C}_1 \| \mathscr{C}_2 = \big((\mathscr{C}_1 \times \mathscr{C}_2) \circ \ell\big) \restriction E$ where ℓ is defined as follows

$$\ell(a) = a \quad \ell(\tau) = \tau \qquad \ell(a,\bar{a}) = \ell(\bar{a},a) = \tau \quad \ell(\tau,a) = \ell(a,\tau) = 0 \qquad \ell(a,\bar{b}) = \ell(\bar{b},a) = 0$$

and for $(\mathscr{C}_1 \times \mathscr{C}_2) \circ \ell = \langle E', C', \ell' \rangle$ we have the set $E = \{e \in E' \mid \ell'(e) \neq 0\}$.

In configuration structures $\langle E, C, \ell \rangle$ we denote $x \xrightarrow{e} x'$ the configurations $x, x' \in C$ such that $x = x' \cup \{e\}$ and with $x' \overset{e}{\rightsquigarrow} x$ the symmetric relation. We use $x' \overset{e}{\twoheadrightarrow} x$ for either $x \xrightarrow{e} x'$ or $x' \overset{e}{\rightsquigarrow} x$: if $\ell(e) = \alpha$, we sometimes write $x \overset{\alpha}{\twoheadrightarrow} x'$.

Definition 8 (Partial order)**.** Let $x \in C$ and $e_1, e_2 \in x$. Then $e_1 \leq_x e_2$ iff $\forall x_2 \in C, x_2 \subseteq x, e_2 \in x_2 \implies e_1 \in x_2$.

If $e_1 \leq_x e_2$, we say that e_1 *happens before* e_2 or that e_1 causes e_2 in the configuration x. Morphisms on configuration structures reflect causality: if $\pi : \mathscr{C}_1 \to \mathscr{C}_2$ and for $e_1, e_2 \in x$ and $x \in C_1$, if $\pi(e_1) \leq_{\pi(x)} \pi(e_2)$ then $e_1 \leq_x e_2$.

Definition 9 (Substructure)**.** $\langle E_1, C_1, \ell_1 \rangle \subseteq \langle E_2, C_2, \ell_2 \rangle$ iff $E_1 \subseteq E_2, C_1 \subseteq C_2$ and $\ell_1 = \ell_2|_{E_1}$.

3 Encoding RCCS in configuration structures

We start by encoding a CCS term into configuration structures and show an operational correspondence between the term and its encoding. Intuitively, the configuration structure of a process without memory depicts all its possible future behaviour. We also introduce a notion of context for configuration structures. Then we proceed to encode a RCCS term. A reversible process can do backward transitions but only up to a point: until it reaches the empty memory. We encode then a RCCS terms as an "address" in the configuration structure of its origin. This allows us to encode both the past and the future of a process in the same configuration structure. However the syntax of a process is not informative enough, hence we restrict the encoding to a class of processes. Lastly we show an operational correspondence for RCCS terms and their encoding.

3.1 Encoding CCS

We start by encoding a term without memory, that is a CCS term. We do so by structural induction on the term using the operations defined previously (Definition 7):

$$[\![P_1 | P_2]\!] = [\![P_1]\!] | [\![P_2]\!] \qquad [\![P_1 + P_2]\!] = [\![P_1]\!] + [\![P_2]\!] \qquad [\![\alpha.P]\!] = \alpha.[\![P]\!] \qquad [\![\nu a.P]\!] = [\![P]\!] \restriction E_a$$

Note that this encoding and its correspondence with CCS was first proposed by Winksell [12].

To show a strong bisimulation between a CCS process and its encoding, we introduce the following transformation of a configuration structures representing, intuitively, the structure we obtain after a transition: $\langle E, C, \ell \rangle \setminus x = \langle E', C', \ell \restriction E' \rangle$ with $E' = \cup C'$ and $x' \in C' \iff \exists y \in C, x \subseteq y$ and $x' = y \setminus x$.

Intuitively, $\mathscr{C} \setminus x$ is the configuration resulting from the suppression of the events of x in all configurations of $\mathscr{C}$. We call *minimal* (with respect to the partial order in Definition 8) an event whose singleton is a configuration.

Proposition 1. *Let x be a configuration in $\mathscr{C}$, then $\mathscr{C} \setminus x$ is a configuration structure.*

Proposition 2 (Strong bisimulation between a CCS process P and $[\![P]\!]$)**.**

$$\text{If } P \xrightarrow{\alpha} Q \text{ then } \exists e \in [\![P]\!] \text{ minimal s.t. } \ell(e) = \alpha \text{ and } [\![Q]\!] = [\![P]\!] \setminus \{e\} \qquad \text{(Soundness)}$$

$$\forall e \in [\![P]\!] \text{ minimal, } \exists Q \text{ s.t. } P \xrightarrow{\ell(e)} Q \text{ and } [\![Q]\!] = [\![P]\!] \setminus \{e\} \qquad \text{(Completeness)}$$

Proof. We show this by induction on the derivation $P \xrightarrow{\alpha} Q$ for (Soundness) and by structural induction on $[\![P]\!]$ for (Completeness). □

We cannot define a notion of context for configuration structures in general, as it is not clear what a configuration structure with a hole would be. However, if a configuration structure $\mathscr{C}$ has an operational meaning, i.e. if $\exists P$ a CCS process such that $\mathscr{C} = [\![P]\!]$, we can use a CCS context $C[\cdot]$ that we instantiate with P.

When reasoning on contexts in CCS, it is common to distinguish between the part of a transition fired by the context alone and the part fired by the process. In the operational setting, one can easily decompose the term $C[P]$ thanks to the rules of the LTS. We need a similar reasoning for the term $[\![C[P]]\!]$, hence we attach to the context $C[\cdot]$ and process P a projection morphism $\pi_{C,P} : [\![C[P]]\!] \to [\![P]\!]$ that can retrieve the parts of a configuration in $[\![C[P]]\!]$ that belong to $[\![P]\!]$[1].

Morphisms do not preserve causality in general. In the case of a product we can show that all causalities are due to one of the two configuration structures.

Proposition 3. *Let $x \in \mathscr{C}_1 \times \mathscr{C}_2$. Then $e <_x e' \iff$ either $\pi_1(e) <_{\pi_1(x)} \pi_1(e')$ or $\pi_2(e) <_{\pi_2(x)} \pi_2(e')$.*

Without much difficulty the result can be extended to say that in $[\![C[P]]\!]$, causality appears due to either the causality in $C[\cdot]$ or the causality in P: a context can add but cannot remove causality in the process [1].

3.2 Encoding RCCS

A RCCS term corresponds to a configuration in the configuration structure of its origin. We can use the past execution, that is the memory of R to point to a configuration but it is not discriminatory enough. Consider the process $\emptyset \rhd a.0 + a.b.0 \xrightarrow{a} R$ whose configuration structure is $\mathscr{C}_3$ in Figure 3. To determine which of the configurations labelled a correspond to R we have to consider the future of R as well.

Hence we choose a configuration that respects the past and the future of R, but this is still not enough. Consider the process $a.b.0 + a.b.0$ whose configuration is $\mathscr{C}_2$ in Figure 3. For the trace $\emptyset \rhd a.b + a.b \xrightarrow{\alpha} b$ there is no way to choose between the two configurations labelled a. From now on, we consider only RCCS processes for which the underlying CCS process has the property that $\text{collapse}(P) = P$, where collapse is defined below.

Definition 10 (Collapse)**.**

$$\begin{aligned}
\text{collapse}(\alpha.P + \alpha.Q) &= \alpha.\text{collapse}(P), \text{ if } \text{collapse}(P) = \text{collapse}(Q) & \text{collapse}(\alpha.P) &= \alpha.\text{collapse}(P)\\
\text{collapse}(\alpha.P + \beta.Q) &= \alpha.\text{collapse}(P) + \beta.\text{collapse}(Q) & \text{collapse}((a)P) &= (a)\,\text{collapse}(P)\\
\text{collapse}(\alpha.P | \alpha.Q) &= \alpha.\text{collapse}(P), \text{ if } \text{collapse}(P) = \text{collapse}(Q) & \text{collapse}(0) &= 0\\
\text{collapse}(P|Q) &= \text{collapse}(P) | \text{collapse}(Q)
\end{aligned}$$

[1] The formal definitions and the missing proofs can be found in Appendix A.

Hence each process points to a unique configuration, enabling us to encode the past behaviour without difficulty. Thus we define an "address" function that, given the configuration structure of the process's origin and a trace to the process we want to encode, returns the configuration corresponding to the current state.

Definition 11 (Encoding RCCS processes in configuration structures)**.** Given R a RCCS process, its encoding $[\![R]\!]$ is defined as the couple $([\![O_R]\!], \mathrm{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star R))$, where

$$\mathrm{ad}_{[\![O_R]\!]}(x, R_1 \xrightarrow{\alpha} R_2 \longrightarrow^\star R_3) = \mathrm{ad}_{[\![O_R]\!]}(x \cup \{e\}, R_2 \longrightarrow^\star R_3)) \quad \text{if } \begin{cases} x \cup \{e\} \in [\![O_R]\!] \\ \text{and} \\ [\![\varepsilon(R_2)]\!] \subseteq \left([\![O_R]\!] \setminus (x \cup \{e\})\right) \end{cases}$$

$$\mathrm{ad}_{[\![O_R]\!]}(x, R_2 \longrightarrow^\star R_3)) = x \qquad \text{if } R_2 = R_3$$

Let us show that the encoding is correct, and in particular that the function ad is well defined.

Proposition 4 (Soundness of the RCCS encoding)**.** *Let R be a process, then $\exists! x \in [\![O_R]\!]$ such that* $\mathrm{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star R) = x$.

The proof, presented in Appendix A, proceeds by induction on the trace, uses Proposition 2 and the collapsing hypothesis (Definition 10).

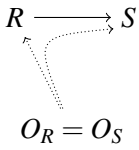
Let us now define a transition relation on configuration structures, useful in showing the operational correspondence between terms of RCCS and their encoding.

Definition 12 (Transition in configuration structures)**.** Define $([\![P]\!], x) \xrightarrow{\ell(e)} ([\![P]\!], x \cup \{e\})$ for $x \cup \{e\} \in [\![P]\!]$.

Lemma 2 (Operational correspondence)**.** *1. if $R \overset{i:\alpha}{\twoheadrightarrow} S$ then $[\![R]\!] \overset{\alpha}{\twoheadrightarrow} [\![S]\!]$;*

2. let $[\![R]\!] = (\mathscr{C}, x)$; if $(\mathscr{C}, x) \xrightarrow{\ell(e)} (\mathscr{C}, x \cup \{e\})$ then $\exists S$, such that for some $i \in \mathsf{I}$ fresh, $R \overset{i:\alpha}{\twoheadrightarrow} S$ and $[\![S]\!] = (\mathscr{C}, x \cup \{e\})$.

Proof. 1. As $R \xrightarrow{i:\alpha} S$, $O_R = O_S$ and we are in the following situation:



We have that $[\![S]\!] = ([\![O_R]\!], x_s)$, where $x_s = \mathrm{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star S) = \mathrm{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star R \xrightarrow{\alpha} S) = x_R \cup \{e\}$. As $[\![R]\!] = ([\![O_R]\!], x_R)$ it follows that $([\![O_R]\!], x_R) \xrightarrow{\alpha} ([\![O_R]\!], x_S)$. The proof for the backward direction is similar except that it uses the trace up to R.

2. From $(\mathscr{C}, x) \xrightarrow{\ell(e)} (\mathscr{C}, x \cup \{e\})$ we have that $x \cup \{e\} \in \mathscr{C}$. Then $\{e\} \in \mathscr{C} \setminus x$. From $[\![R]\!] = (\mathscr{C}, x)$ we have that $\mathscr{C} \setminus x = [\![\varepsilon(R)]\!]$, hence $\{e\} \in [\![\varepsilon(R)]\!]$. We use Proposition 2 and obtain that $\exists P$ such that $\varepsilon(R) \xrightarrow{\ell(e)} P$. Then due to the strong bisimulation between a RCCS term and its corresponding CCS term [2], we have that, for some i, $R \overset{i:\alpha}{\twoheadrightarrow} S$. That $[\![S]\!] = (\mathscr{C}, x \cup \{e\})$ follows from a similar argument as in 1. above. □

4 Definition of Bisimulations

In this section we adapt to configuration structures the definitions of barb and strong back-and-forth barbed bissimulation on RCCS terms (Definition 3 and Definition 4). We define hereditary history preserving

bisimulation, show that they "translate" the sister notion on RCCS terms (Lemma 3), and use two family of relations, denoted F_i and B_i, to inductively approximate the bisimulation (Lemma 4).

Definition 13. A *strong back-and-forth barbed bisimulation on labelled configuration structures* is a symmetric relation $\mathscr{R} \subseteq C_1 \times C_2$ such that $(\emptyset,\emptyset) \in \mathscr{R}$, and if $(x_1,x_2) \in \mathscr{R}$, then

$$x_1 \overset{e_1}{\rightsquigarrow} x_1' \implies \exists x_2' \in C_2 \text{ s.t. } x_2 \overset{e_2}{\rightsquigarrow} x_2', \text{ with } \ell_1(e_1) = \ell_2(e_2) = \tau \text{ and } (x_1',x_2') \in \mathscr{R}; \quad \text{(Back)}$$

$$x_1 \xrightarrow{e_1} x_1' \implies \exists x_2' \in C_2 \text{ s.t. } x_2 \xrightarrow{e_2} x_2', \text{ with } \ell_1(e_1) = \ell_2(e_2) = \tau \text{ and } (x_1',x_2') \in \mathscr{R}; \quad \text{(Forth)}$$

$$\text{if } \exists e_1 \in E_1 \text{ s.t. } \ell_1(e_1) \neq \tau \text{ and } x_1 \xrightarrow{e_1} x_1' \text{ then } \exists x_2' \in C_2 \text{ s.t. } x_2 \xrightarrow{e_2} x_2', \text{ with } \ell_1(e_1) = \ell_2(e_2). \quad \text{(Barbed)}$$

Let $\mathscr{C}_1 \dot{\sim}^\tau \mathscr{C}_2$ if and only if there exists a strong back-and-forth barbed bisimulation between $\mathscr{C}_1$ and $\mathscr{C}_2$.

Denote $\sim^\tau$ a symmetric relation on terms that have an operational meaning such that if $[\![P_1]\!] \sim^\tau [\![P_2]\!]$ then $\forall C$, $[\![C[P_1]]\!] \dot{\sim}^\tau [\![C[P_2]]\!]$.

Let us now show that the relation in Definition 13 is the relation induced by the barbed congruence on processes.

Lemma 3. *$R \sim^\tau S \implies [\![\varepsilon(O_R)]\!] \sim^\tau [\![\varepsilon(O_S)]\!]$ and $[\![P]\!] \sim^\tau [\![Q]\!] \implies P \sim^\tau Q$.*

Proof. Both case are similar :

$$\begin{aligned}
R \sim^\tau S &\implies O_R \sim^\tau O_S && \text{(By Lemma 1)}\\
&\implies \varepsilon(O_R) \sim^\tau \varepsilon(O_S) && \text{(As } \varepsilon(\emptyset \rhd P) = P\text{)}\\
&\implies \forall C[\cdot], C[\varepsilon(O_R)] \dot{\sim}^\tau C[\varepsilon(O_S)] && \text{(By Definition 4)}\\
&\implies \forall C[\cdot], [\![C[\varepsilon(O_R)]]\!] \dot{\sim}^\tau [\![C[\varepsilon(O_S)]]\!] && \text{(By the Soundness part of Proposition 2)}\\
&\implies [\![\varepsilon(O_R)]\!] \sim^\tau [\![\varepsilon(O_S)]\!] && \text{(By Definition 13)}
\end{aligned}$$

□

Definition 14. A *hereditary history preserving bisimulation on labelled configuration structures* is a symmetric relation $\mathscr{R} \subseteq C_1 \times C_2 \times \mathscr{P}(E_1 \times E_2)$ such that $(\emptyset,\emptyset,\emptyset) \in \mathscr{R}$ and if $(x_1,x_2,f) \in \mathscr{R}$, then

$$f \text{ is a label and order preserving bijection between } x_1 \text{ and } x_2$$

$$x_1 \xrightarrow{e_1} x_1' \implies \exists x_2' \in C_2 \text{s.t. } x_2 \xrightarrow{e_2} x_2' \text{ and } f = f' \restriction x_1, (x_1',x_2',f') \in \mathscr{R}$$

$$x_1 \overset{e_1}{\rightsquigarrow} x_1' \implies \exists x_2' \in C_2 \text{s.t. } x_2 \overset{e_2}{\rightsquigarrow} x_2' \text{ and } \textit{and} f' = f \restriction x_2, (x_1',x_2',f') \in \mathscr{R}$$

We define *bisimilarity*, denoted $\mathscr{C}_1 \sim \mathscr{C}_2$, as the greatest hereditary history preserving bisimulation on labelled configuration structures.

Note that $\mathscr{C}_1 \sim \mathscr{C}_2$ is an abuse of notation as $\sim$ is a relation defined on $C_1 \times C_2 \times \mathscr{P}(E_1 \times E_2)$. Due to the restrictions imposed on the configuration structures (see Remark 1) there is a unique mapping between events for the greatest hhp bisimulation.

We can give an inductive characterisation of HHPB by reasoning on the structures up to a level: we ignore the configurations that have greater cardinality then the considered level. Hhpb is then the relation obtained when we reach the top level. Hence we can detect, whenever two configuration structures are not hhp bisimilar, at which level the bisimulation does no longer hold. We do this with the aid of the two following functions.

Definition 15 (F_i, B_i)**.** Given $\mathscr{C}_1$, $\mathscr{C}_2$ two configuration structures, we let k be the cardinal of the largest configuration of $\mathscr{C}_1$[2] and define, for all $x_1 \in C_1$, $x_2 \in C_2$ and f a label and order-preserving function:

$$(x_1,x_2,f) \in F_i \Leftrightarrow \begin{cases} \text{Card}(x_1) = \text{Card}(x_2) = i,\ f \text{ any label and order-preserving function} & \text{if } i = k \\ \forall x_1', \exists x_2', x_1 \xrightarrow{e_1} x_1', x_2 \xrightarrow{e_1} x_2' \text{ and } f = f' \restriction x_1 \text{ s.t. } (x_1', x_2', f') \in F_{i+1} & \text{elsewhere} \end{cases}$$

$$(x_1,x_2,f) \in B_i \Leftrightarrow \begin{cases} (x_1,x_2,f) \in F_i & \text{if } i = 0 \\ \forall x_1', \exists x_2', x_1 \overset{e_1}{\rightsquigarrow} x_1', x_2 \overset{e_1}{\rightsquigarrow} x_2' \text{ and } f' = f \restriction x_2 \text{ s.t. } (x_1', x_2', f') \in F_{i-1} \cap B_{i-1} & \text{elsewhere} \end{cases}$$

The relation B_i is built on top of F_i: it "tests for the backward steps" all the couples that "passed the forward test". It should be remarked that, with this definition, $B_i \subseteq F_i$, but, at the price of slight modifications, one could define F_i on top of B_i.

Example 4.1. Consider $\mathscr{C}_3$ and $\mathscr{C}_4$ of Figure 3, the relations F_n are enough to discriminate them:

$$F_2 = \big(\{e_3, e_3'\}, \{e_4, e_4'\}\big); \big(\{e_3, e_3'\}, \{e_4'', e_4'''\}\big) \qquad F_1 = \big(\{e_3\}, \{e_4\}\big); \big(\{e_3\}, \{e_4''\}\big) \qquad F_0 = \emptyset$$

This intuitively is due to the fact that forward transitions are enough to discriminate $a + a.b$ and $a.b + a.b$. However for comparing the processes $a \mid b$ and $a.b + b.a$ whose configurations are $\mathscr{C}_1$ and $\mathscr{C}_2$ of Figure 3, we need the backward moves as well:

$$F_2 = \big(\{e_1, e_1'\}, \{e_2, e_2'\}\big); \big(\{e_1, e_1'\}; \{e_2'', e_2'''\}\big) \qquad F_1 = \big(\{e_1\}, \{e_2\}\big); \big(\{e_1'\}; \{e_2''\}\big) \qquad F_0 = \big(\emptyset, \emptyset\big)$$
$$B_2 = \emptyset \qquad B_1 = \big(\{e_1\}, \{e_2\}\big); \big(\{e_1'\}; \{e_2''\}\big) \qquad B_0 = F_0 = \big(\emptyset, \emptyset\big)$$

The following proposition states that pairs of configurations are in a bisimulation relation if they have the same cardinality. It follows from the fact that any configuration is reachable from the empty set and that they have to mimic each other's step in the backward direction.

Proposition 5. *Let $\mathscr{C}_1 \sim \mathscr{C}_2$ and $x_1 \in \mathscr{C}_1$, $x_2 \in \mathscr{C}_2$. If $\exists f$ such that $(x_1,x_2,f) \in \{\sim\}$ then* $\text{Card}(x_1) = \text{Card}(x_2)$.

We are going to prove a fine lemma that will be handy to prove Theorem 1. It implies that if for all $n \leqslant k$ the maximal cardinal considered, $F_n \cap B_n \neq \emptyset$, then $\cup_{n \leqslant k}(F_n \cap B_n)$ is a bisimulation.

Lemma 4. *For all $\mathscr{C}_1$, $\mathscr{C}_2$, if $\mathscr{C}_1 \sim \mathscr{C}_2$, then $\forall x_1 \in \mathscr{C}_1 (\exists x_2 \in \mathscr{C}_2, \exists f, (x_1,x_2,f) \in F_n \cap B_n) \iff (\exists x_2 \in \mathscr{C}_2, \exists f, (x_1,x_2,f) \in \sim)$.*

Proof. Let us denote $\mathscr{R}$ the relation $\sim$. One should first remark that $\mathscr{C}_1 \sim \mathscr{C}_2$ implies that $\forall x_1 \in \mathscr{C}_1, \exists x_2 \in \mathscr{C}_2$, and $\exists f$ such that $(x_1,x_2,f) \in \mathscr{R}$, as $(\emptyset,\emptyset,\emptyset) \in \mathscr{R}$ and all configurations are reachable from the empty set. The reader should notice that the $x_2 \in \mathscr{C}_2$ and f on both sides of the $\iff$ symbols may be different.

We prove that statement by induction on the cardinal of x_1.

$\text{Card}(x_1) = 0$

- $\Rightarrow$ $x_2 \in \mathscr{C}_2$ s.t. $(\emptyset,x_2,f) \in \mathscr{R}$ follows by the definition of the bisimulation from $x_2 = \emptyset$ and $f = \emptyset$.
- $\Leftarrow$ By definition, $F_0 \cap B_0 = F_0$. Since there exists $x_2 \in \mathscr{C}_2$ such that $(\emptyset,x_2,f) \in \mathscr{R}$, we know that any forward transition made by $\emptyset$ can be simulated by a forward transition from x_2, and that the elements obtained are in the relation $\mathscr{R}$. By an iterated use of this notion, we can find "maximal" elements $x_1^m \in \mathscr{C}_1$ and $x_2^m \in \mathscr{C}_2$ (that is, elements of maximal cardinality, k) such that $(x_1^m, x_2^m, f^m) \in \mathscr{R}$. By Proposition 5, x_1^m and x_2^m have the same cardinality, and $(x_1^m, x_2^m, f^m) \in F_k$. By just "reversing the trace", we can go backward and stay in relation F_i until $i = 0$, hence we found the x_2 and f we were looking for.

[2] All the configurations we manipulate here are finite. In an infinite setting, this bound can be viewed as a way to define an "up to k steps bisimulation".

$\mathrm{Card}(x_1) = k+1$ As $\mathrm{Card}(x_1) > 0$, we know there exists x_1' such that $x_1 \overset{e_1}{\rightsquigarrow} x_1'$.

$\Rightarrow$ Let x_2 and f such that $(x_1, x_2, f) \in F_{k+1} \cap B_{k+1}$. We know that

$$\forall x_1', \exists x_2' \text{ and } f', x_1 \overset{e_1}{\rightsquigarrow} x_1', x_2 \overset{e_1}{\rightsquigarrow} x_2' \text{ and } (x_1', x_2', f') \in B_k \qquad \text{(By Definition of } B_k\text{)}$$
$$\exists x_2'', f'', (x_1', x_2'', f'') \in \mathscr{R} \qquad \text{(By Induction Hypothesis)}$$

And as $x_1' \xrightarrow{e_1} x_1$, there exists x_2''' and f''' such that $(x_1, x_2''', f''') \in \mathscr{R}$.

$\Leftarrow$ We prove it by contraposition: suppose that $\exists x_2, f$ such that $(x_1, x_2, f) \in \mathscr{R}$, we prove that $\forall x_2$, $(x_1, x_2, f) \notin F_{k+1} \cap B_{k+1}$ leads to a contradiction.

As $(x_1, x_2, f) \in \mathscr{R}$, $\exists x_1', x_2', f'$ such that $x_1 \overset{e_1}{\rightsquigarrow} x_1'$, $x_2 \overset{e_1}{\rightsquigarrow} x_2'$ and $(x_1', x_2', f') \in \mathscr{R}$. By induction hypothesis, $\exists x_2''$ and $\exists f''$ such that $(x_1', x_2'', f'') \in F_k \cap B_k$. As $x_1' \xrightarrow{e_1} x_1$, $\exists x_2'''$ and $\exists f'''$ such that $x_2'' \xrightarrow{e_1} x_2'''$ and $(x_1, x_2''', f''') \in F_{k+1}$, by definition of F_k.

So $(x_1, x_2''', f''') \notin B_{k+1}$, but as $x_1 \overset{e_1}{\rightsquigarrow} x_1'$ and $x_2''' \overset{e_1}{\rightsquigarrow} x_2''$, and as moreover $(x_1', x_2'', f'') \in F_k \cap B_k$, we have that $(x_1, x_2''', f''') \in B_{k+1}$.

From this contradiction we know that we found the right element (x_2''') that is in relation with x_1 according to $F_{k+1} \cap B_{k+1}$. □

5 Correspondence between HHPB and Strong Barbed Congruence

In this section we use the relations defined in Sect. 4 to show that two processes are barbed congruent whenever their denotations are in the HHPB relation (Theorem 1). One direction is straightforward (Proposition 6), whereas the other is more technical and, as in CCS [7], follows by contradiction. It uses the relations F_i and B_i (Definition 15) to build contexts that discriminate processes that are not bisimilar.

Remark 1 (On auto-concurrency and others limitations)**.** In the proofs that follow we need to uniquely identify configurations based on the labels and orders of the "open" (i.e. non synchronized) events. This is not possible in processes as $a.P \mid a.Q$ or $a.P + a.Q$. Auto concurrency [4, Definition 9.5] forbids the first kind of processes. But we need a stronger condition, a sort of auto conflict, to forbid the second, that is not ruled out by the collapse function (Definition 10). Hence in the following we do not consider processes that exhibit auto concurrency *or* auto conflict.

The problem is specific to the encoding in configuration structures. It appears in the encoding of Winskel [12], and is treated thanks to *tags* that discriminates between the right- and the left-hand side of the sum and of the product [13]. Hence we can retrieve the whole class of processes by adding more information on the labels, at the cost of a more cumbersome presentation.

Proposition 6. $[\![P_1]\!] \sim [\![P_2]\!] \implies \forall C, [\![C[P_1]]\!] \sim [\![C[P_2]]\!]$

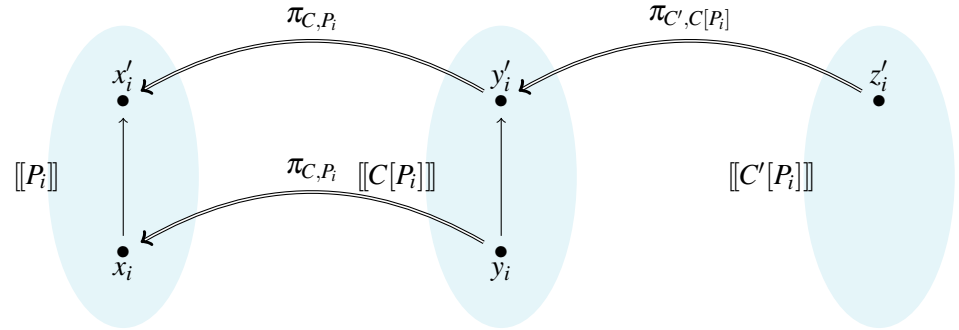
The proof, exposed in Appendix A, amounts to carefully build a relation between $[\![C[P_1]]\!]$ and $[\![C[P_2]]\!]$ that reflects the known bissimulation between $[\![P_1]\!]$ and $[\![P_2]\!]$. Its uses that causality in a product is the result of the entanglement of the causality of its elements (Proposition 3).

Theorem 1. $[\![P_1]\!] \sim [\![P_2]\!] \iff [\![P_1]\!] \sim^{\tau} [\![P_2]\!]$

Proof. The left-to-right direction follows from the definition of $\sim$ (Definition 14) and from Proposition 6.

We prove the other direction by contraposition: let us suppose that $[\![P_1]\!] \sim^{\tau} [\![P_2]\!]$ and $[\![P_1]\!] \not\sim [\![P_2]\!]$, we will find a contradiction. Figure 4 presents the general shape of the configurations at the end of the proof.

For $i \in \{1,2\}$, we have:



We start with $y_1 \sim^\tau y_2$, then prove that $z'_1 \sim^\tau z'_2$, to end up with $(x'_1, x'_2, f) \in F_n \cap B_n$.

Figure 4: Configurations Structures by the end of the proof of Theorem 1

As $[\![P_1]\!] \not\sim [\![P_2]\!]$, by Lemma 4, there exists $x_1 \in [\![P_1]\!]$ such that $\forall x_2 \in [\![P_2]\!]$, $(x_1, x_2, f) \notin F_n \cap B_n$ holds. Note that we can only consider x_2 such that $\mathrm{Card}(x_1) = \mathrm{Card}(x_2) = n$, and that we use the projections $\pi_{C,P}$ (Definition 16) to separate the events of the process P from the events of the context C.

Let us show that for any x_1 we can define $C[\cdot] := \prod_{e_i \in x_i} (\overline{\ell(e_i)} + c_{e_i}) | [\cdot]$ where $c_{e_i} \notin \mathsf{N}(P_1) \cup \mathsf{N}(P_2)$, such that the following holds

- $\exists y_1 \in [\![C[P_1]]\!]$ such that y_1 is closed, $\pi_{C,P_1}(y_1) = x_1$ and $y_1 \not\downarrow c_{e_i}$ for all $e_i \in x_1$;
- We supposed that $[\![P_1]\!] \sim^\tau [\![P_2]\!]$, so $[\![C[P_1]]\!] \sim^\tau [\![C[P_2]]\!]$. Hence $\exists y_2 \in [\![C[P_2]]\!]$ such that $(y_1, y_2, g) \in \sim^\tau$ and $y_2 \not\downarrow c_{e_i}$ for all $e_i \in x_1$.

Moreover we show that $(x_1, \pi_{C,P_1}(y_2), f) \in F_n$, for some f a label and order preserving bijection.

Let us start by showing that such an f exists.

We denote $\pi_{C,P_1}(y_2)$ with x_2. We have that $\forall e_1, e'_1 \in x_1$, and $e_2 \in x_2$,

$$e_2 \in x_2 \iff e_1 \in x_2 \text{ and } \ell(e_1) = \ell(e_2) \tag{1}$$

$$e_1 <_{x_1} e'_1 \implies \pi^{-1}_{C,P_1}(e_1) <_{y_1} \pi^{-1}_{C,P_1}(e'_1) \tag{2}$$

$$\implies g(\pi^{-1}_{C,P_1}(e_1)) <_{y_2} g(\pi^{-1}_{C,P_1}(e'_1)) \tag{3}$$

Remark that (1) follows from $y_2 \not\downarrow c_{e_i}$ and from the fact that if y_1 is closed we can show by contradiction that y_2 is closed as well. Secondly, (2) follows from Proposition 3 and from the form of the context, which does not induce any causality between the events. Lastly, (3) follows from g being an order preserving bijection between y_1 and y_2.

We proceed by induction to show that $(x_1, x_2, f) \in F_n$.

- If $n = k$ for k the maximal cardinal of events in $[\![P_1]\!]$. This case is trivial, as $\mathrm{Card}(x_1) = \mathrm{Card}(x_2) = k$.
- If $n = k-1$ for $k > 1$, we prove that $(x_1, x_2, f) \notin F_{k-1}$ leads to a contradiction. There are two cases:

$$\nexists x'_1, x_1 \xrightarrow{e_1} x'_1, \exists x'_2, x_2 \xrightarrow{e_2} x'_2 \tag{4}$$

$$\exists x'_1, x_1 \xrightarrow{e_1} x'_1, \forall x'_2, x_2 \xrightarrow{e_2} x'_2 \text{ and } (x'_1, x'_2, f') \notin F_k \tag{5}$$

The implication (4) is easier: if $\exists x'_2, x_2 \xrightarrow{e_2} x'_2$, then, as a context cannot remove transitions from the original process, $\exists y'_2, y_2 \xrightarrow{(e_2,\star)} y'_2$. As $[\![C[P_2]]\!] \dot{\sim}^\tau [\![C[P_1]]\!]$, $\exists y'_1, y_1 \xrightarrow{(e_1,\star)} y'_1$, and a similar argument on the context shows that $\exists x'_1, x_1 \xrightarrow{e_1} x'_1$. Hence a contradiction.

To prove (5) requires more work and uses the induction hypothesis. First, let $C'[\cdot] := C[\cdot] | (\overline{\ell(e_1)} + c_{e_1})$. By induction hypothesis, there exists $z'_1 \in [\![C'[P_1]]\!]$ such that z'_1 is closed, $\pi_{C',C[P_1]}(z'_1) = y'_1$ and $z'_1 \not\Downarrow c_{e_i}$ and $z'_1 \not\Downarrow c_{e_1}$ for all $e_i \in x_1$.

By hypothesis, $[\![P_1]\!] \sim^\tau [\![P_2]\!]$, hence $[\![C'[P_1]]\!] \dot{\sim}^\tau [\![C'[P_2]]\!]$ implies that $\exists z'_2, h'$ such that $z_2 \in [\![C'[P_2]]\!]$ and $z'_2 \not\Downarrow c_{e_i}$ and $z'_2 \not\Downarrow c_{e_1}$ for all $e_i \in x_1$.

Let us denote the projection $\pi_{C',C[P_2]}(z'_2)$ as y''_2. As z'_1 is closed, so is z'_2. We can infer using the fact that z'_2 is closed and that $z'_2 \not\Downarrow c_{e_1}$ that $\exists e''_2 \in y''_2$ such that $\ell(e''_2) = \ell(e_1)$ and $y''_2 \setminus \{e''_2\}$ is closed.

From $z'_2 \not\Downarrow c_{e_i}$ we have that $y''_2 \not\Downarrow c_{e_i}$. As there exists a label and order preserving bijection h' between z'_1 and z'_2, and as we forbid auto concurrency and "ambiguous" non deterministic sum (Remark 1), we conclude that $\pi_{C,P_2}(y''_2) = x'_2$ and $\pi_{C',P_2}(z'_2) = x'_2$.

Then we have $\pi_{C',P_1}(z'_1) = x'_1, \pi_{C',P_2}(z'_2) = x'_2$, and by induction hypothesis, $(x'_1, x'_2, f) \in F_k$. But as $x_1 \xrightarrow{e_1} x'_1$ and $x_2 \xrightarrow{e_2} x'_2$, we have that $(x_1, x_2, f) \in F_{k-1}$, hence a contradiction.

To prove that $(x_1, x_2, f) \in B_n$, we use induction, the base case ($n = 0$) being trivial. The step case goes along the line of (and uses) the proof that $(x_1, x_2, f) \in F_n$. □

Conclusion

We showed that, for a restricted class of RCCS processes (without recursion, auto-concurrency or auto-conflict) hereditary history preserving bisimilation has a contextual characterisation in CCS. We used the barbed congruence defined on RCCS as the congruence of reference, adapted it to configuration structures and then showed a correspondence with HHPB. As a proof tool, we defined two inductively relations that approximate HHPB. Consequently we have that adding reversibility into the syntax helps in retrieving some of the discriminating power of configuration structures.

This work follows notable efforts [9, 6] to understand equivalences for reversible processes. There are many interesting continuations. A first one as suggested in the introduction, is to move to weak equivalences, which ignore silent moves τ and focus on the observable part of a process. This is arguably a more interesting relation than the strong one, in which processes have to mimic each other's silent moves. Even if such a relation on configuration structures exists [11] one still has to show that this is indeed the relation we expect. In the denotational setting, the adjective "weak" has sometimes [10, 4] a different meaning: it stands for the ability to change the label and order preserving bijection as the relation grows, to modify choices that were made before this step.

The relations defined so far simulate forward (resp. backward) transitions only with forward (resp. backward) transitions, and only consider "forward" barb. Ignoring the direction of the transitions could introduce some fruitful liberality in the way processes can simulate each other. Depending on the answer, $a + \tau.b$ and $a + b$ would be weakly bisimilar or not. Moreover one can also consider irreversible moves and understand what are the meaningful equivalences in the setting of transactions [3].

Context—which plays a major part in these equivalences—raises questions on the memory handling of RCCS: what about context that could "fix the memory" of an incoherent process? For instance, $C = \langle 1, a, 0\rangle \rhd P' | [\cdot]$ and $P = \langle 1, \bar{a}, 0\rangle \rhd P''$ are incoherent, but $C[P]$ is coherent and can backtrack.

One can easily retrieve auto concurrency and auto conflict by tagging the transitions. Bisimulations have then to consider the tags. Maybe of less interest but important for the generality of these results, one

should include infinite processes as well. This needs a rework of the relations in Definition 15 used to approximate the HHPB.

Acknowledgement We would like to thank D. Varacca and J. Krivine for the very useful discussions as well as the referee for his helpful remarks.

References

[1] Ioana Cristescu, Jean Krivine & Daniele Varacca (2013): *A Compositional Semantics for the Reversible p-Calculus*. In: *LICS*, IEEE Computer Society, pp. 388–397, doi:10.1109/LICS.2013.45.

[2] Vincent Danos & Jean Krivine (2004): *Reversible Communicating Systems*. In Philippa Gardner & Nobuko Yoshida, editors: *CONCUR*, *LNCS* 3170, Springer, pp. 292–307, doi:10.1007/978-3-540-28644-8_19.

[3] Vincent Danos & Jean Krivine (2005): *Transactions in RCCS*. In Martín Abadi & Luca de Alfaro, editors: *CONCUR*, *LNCS* 3653, Springer, pp. 398–412, doi:10.1007/11539452_31.

[4] Robert J. van Glabbeek & Ursula Goltz (2001): *Refinement of actions and equivalence notions for concurrent systems*. *Acta Inform.* 37(4/5), pp. 229–327, doi:10.1007/s002360000041.

[5] André Joyal, Mogens Nielsen & Glynn Winskel (1996): *Bisimulation from Open Maps*. *Inf. Comput.* 127(2), pp. 164–185, doi:10.1006/inco.1996.0057.

[6] Ivan Lanese, Claudio Antares Mezzina & Jean-Bernard Stefani (2010): *Reversing Higher-Order Pi*. In Paul Gastin & François Laroussinie, editors: *CONCUR*, *LNCS* 6269, Springer, pp. 478–493, doi:10.1007/978-3-642-15375-4_33.

[7] Robin Milner & Davide Sangiorgi (1992): *Barbed Bisimulation*. In Werner Kuich, editor: *ICALP*, *LNCS* 623, Springer, pp. 685–695, doi:10.1007/3-540-55719-9_114.

[8] Mogens Nielsen, Gordon D. Plotkin & Glynn Winskel (1981): *Petri Nets, Event Structures and Domains, Part I*. *Theoret. Comput. Sci.* 13, pp. 85–108, doi:10.1016/0304-3975(81)90112-2.

[9] Iain Phillips & Irek Ulidowski (2007): *Reversibility and Models for Concurrency*. *ENTCS* 192(1), pp. 93–108, doi:10.1016/j.entcs.2007.08.018.

[10] Iain Phillips & Irek Ulidowski (2012): *A hierarchy of reverse bisimulations on stable configuration structures*. *MSCS* 22(2), pp. 333–372, doi:10.1017/S0960129511000429.

[11] Walter Vogler (1993): *Bisimulation and Action Refinement*. *Theoret. Comput. Sci.* 114(1), pp. 173–200, doi:10.1016/0304-3975(93)90157-O.

[12] Glynn Winskel (1982): *Event Structure Semantics for CCS and Related Languages*. In Mogens Nielsen & Erik Meineche Schmidt, editors: *ICALP*, *LNCS* 140, Springer, pp. 561–576, doi:10.1007/BFb0012800.

[13] Glynn Winskel & Mogens Nielsen (1995): *Models for concurrency*. In S. Abramsky, Dov M. Gabbay & T. S. E. Maibaum, editors: *Semantic Modelling*, *Handbook of Logic in Computer Science* 4, Oxford University Press, pp. 1–148.

A Appendices

A.1 Additional Definition

Definition 16 (Context for configuration structures)**.** Let P a CCS a process and $C[\cdot]$ a context. Then $\pi_{C,P}$ is as the projection morphism $\pi_{C,P} : [\![C[P]]\!] \to [\![P]\!]$ defined inductively on the structure of $[\![C[P]]\!]$:

- $\pi_{C,P} : [\![\alpha.C'[P]]\!] \to [\![P]\!]$ is defined as $\pi_{C,P}(e) = \pi_{C',P}(e)$;
- $\pi_{C,P} : [\![C'[P]|P']\!] \to [\![P]\!]$ is defined as $\pi_{C,P}(e) = \pi_{C',P}(\pi_1(e))$, where $\pi_1 : [\![C'[P]|P']\!] \to [\![C'[P]]\!]$ is the projection morphism defined by the product;
- $\pi_{C,P} : [\![C'[P] + P']\!] \to [\![P]\!]$ is defined as $\pi_{C,P}(e) = \pi_{C',P}(\pi_1(e))$, where $\pi_1 : [\![C'[P] + P']\!] \to [\![C'[P]]\!]$ is the projection morphism defined by the coproduct;
- $\pi_{C,P} : [\![(a)C'[P]]\!] \to [\![P]\!]$ defined as $\pi_{C,P}(e) = \pi_{C',P}(e)$.

That the projection $\pi_{C,P} : [\![C[P]]\!] \to [\![P]\!]$ is a morphism follows by a simple case analysis. We naturally extend $\pi_{C,P}$ to configurations.

A.2 Proof of Proposition 4

Proof. Without loss of generality, the trace $O_R \longrightarrow^\star R$ can be considered to be only forward: every reversible trace can be re-arranged as a succession of backward moves followed by a succession of forward moves [2, Lemma 10], but O_R cannot go backward. We proceed by induction on the trace $O_R \longrightarrow^\star R$. Let $\mathrm{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star R) = x_n$, for $x_n \in [\![O_R]\!]$ and such that $[\![\varepsilon(R)]\!] = [\![\varepsilon(O_R)]\!] \setminus x_n$. We have to show that

$$\mathrm{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star R \xrightarrow{a} R_{n+1}) = x_n \cup \{e\} \text{ and } x_n \cup \{e\} \in [\![O_R]\!], \text{ with } [\![\varepsilon(R_{n+1})]\!] = [\![O_R]\!] \setminus (x_n \cup \{e\}).$$

We have that $\mathrm{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star R \xrightarrow{a} R_{n+1}) = \mathrm{ad}_{[\![O_R]\!]}(x_n, R \xrightarrow{a} R_{n+1})$ and that $[\![\varepsilon(R)]\!] = [\![O_R]\!] \setminus x_n$. We want to show that for $R \xrightarrow{\alpha} R_{n+1}$,$\exists!\{e\} \in [\![\varepsilon(R)]\!]$ such that $[\![\varepsilon(R_{n+1})]\!] = [\![\varepsilon(R)]\!] \setminus \{e\}$. We consider only the case $\alpha = a$, the rest is similar. We rewrite $R \equiv (b_1 \dots b_n)(m_1 \rhd a.P_1 \mid P_2)$ and $R_{n+1} \equiv (b_1 \dots b_n)(m_1 \rhd P_1 \mid P_2)$ and hence $\varepsilon(R) = (b_1 \dots b_n)(a.P_1 \mid P_2)$ and $\varepsilon(R_{n+1}) = (b_1 \dots b_n)(P_1 \mid P_2)$. We want to show that $\exists! e \in [\![O_R]\!] \setminus x_n$ such that $\ell(e) = \alpha$ and

$$[\![\varepsilon(R_{n+1})]\!] = [\![\varepsilon(O_R)]\!] \setminus (x \cup \{e\}).$$

But $[\![\varepsilon(O_R)]\!] \setminus (x \cup \{e\}) = [\![\varepsilon(R)]\!] \setminus \{e\}$. Hence it is enough to show that $\exists! e \in [\![\varepsilon(R)]\!]$ such that $\ell(e) = \alpha$ and

$$[\![\varepsilon(R_{n+1})]\!] = [\![\varepsilon(R)]\!] \setminus \{e\}$$

which is equivalent to show that

$$[\![(b_1 \dots b_n)(P_1 \mid P_2)]\!] = [\![(b_1 \dots b_n)(a.P_1 \mid P_2)]\!] \setminus \{e\}.$$

From Proposition 2 such an event exists and its uniqueness follows from the collapsing hypothesis (Definition 10).

Let us prove that if $x \in [\![(b_1 \dots b_n)(P_1 \mid P_2)]\!]$ then $x \in [\![(b_1 \dots b_n)(a.P_1 \mid P_2)]\!] \setminus \{e\}$. The other direction is similar. Let us unfold the definition of the encoding. We have the following equalities:

$$[\![(b_1 \dots b_n)(P_1 \mid P_2)]\!] = (b_1 \dots b_n)([\![P_1]\!] \times [\![P_2]\!]) \restriction X$$
$$[\![(b_1 \dots b_n)(a.P_1 \mid P_2)]\!] = (b_1 \dots b_n)(a.[\![P_1]\!] \times [\![P_2]\!]) \restriction Y$$

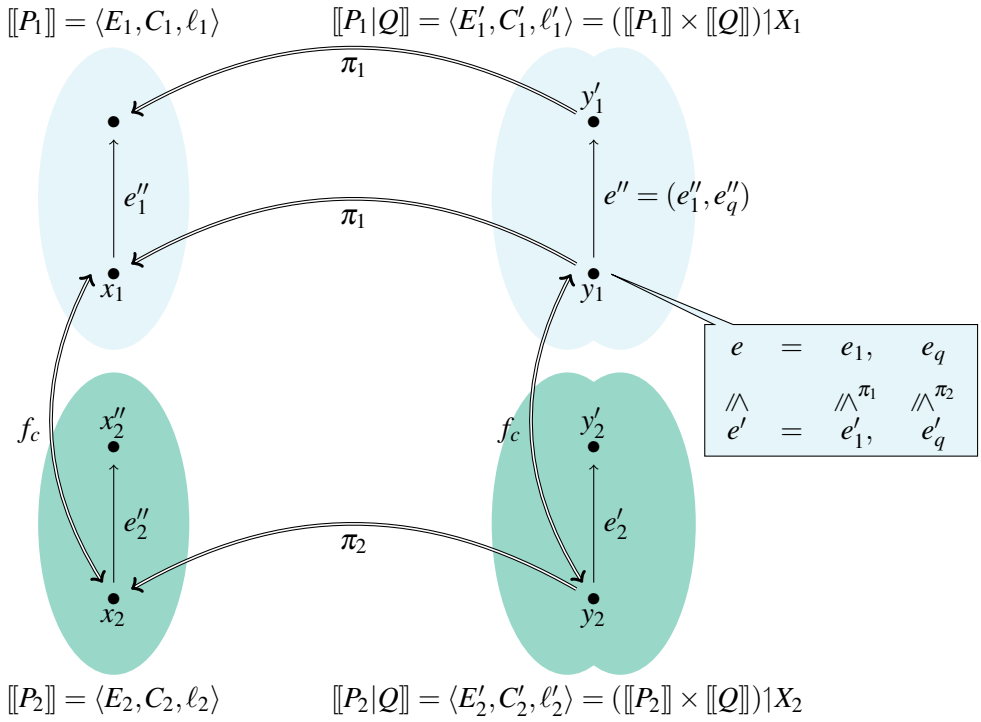


Figure 5: Configurations Structures by the end of the proof of Proposition 6

If $x \in (b_1 \dots b_n)([\![P_1]\!] \times [\![P_2]\!]) \restriction X$ then

$$\nexists e \in x, \ell(e) \in \{b, \bar{b}, 0\}. \tag{6}$$

Hence $x \in ([\![P_1]\!] \times [\![P_2]\!])$. Let π_1, π_2 the two projections defined by the product. Then

$$\pi_1(x) \in [\![P_1]\!] \text{ and } \pi_2(x) \in [\![P_2]\!]. \tag{7}$$

As $\pi_1(x) \in [\![P_1]\!]$, and from the definition of $[\![a.P_1]\!]$ we have that $\exists e_1$, $\ell(e_1) = a$ and such that $\{e_1\} \cup \pi_1(x) \in a.[\![P_1]\!]$. From Equation 7 we have that $\exists x_2 \in a.[\![P_1]\!] \times [\![P_2]\!]$ such that $\pi_1(x_2) = \{e_1\} \cup \pi_1(x)$ and $\pi_2(x_2) = \pi_2(x)$. Hence $\exists! e$ such that $\pi_1(e) = e_1$, $\pi_2(e) = \star$ and $x_2 = \{e\} \cup x$. From Equation 6 we have that $x_2 \in (b_1 \dots b_n)(a.[\![P_1]\!] \times [\![P_2]\!]) \restriction Y$. From the definition of $[\![O_R]\!] \setminus \{e\}$ we infer that if $x \cup \{e\} \in (b_1 \dots b_n)(a.[\![P_1]\!] \times [\![P_2]\!]) \restriction Y$ then $x \in [\![(b_1 \dots b_n)(a.P_1 \mid P_2)]\!] \setminus \{e\}$.

From $[\![\varepsilon(R)]\!] = [\![O_R]\!] \setminus x_n$, we have that $\forall y \in [\![\varepsilon(R)]\!]$, $\exists y \cup x_n \in [\![O_R]\!]$. In particular $x_n \cup \{e\} \in [\![O_R]\!]$. Hence $\text{ad}_{[\![O_R]\!]}(\emptyset, O_R \longrightarrow^\star R \xrightarrow{a} R_{n+1}) = x_n \cup \{e\}$ with $[\![\varepsilon(R_{n+1})]\!] = [\![O_R]\!] \setminus (x_n \cup \{e\})$. □

A.3 Proof of Proposition 6

Proof. We only consider the following case:

$$\forall P_1, P_2, [\![P_1]\!] \sim [\![P_2]\!] \implies \forall Q, [\![P_1|Q]\!] \sim [\![P_2|Q]\!]$$

As $[\![P_1]\!] \sim [\![P_2]\!]$, there exists $\mathscr{R}$ a hereditary history preserving bisimulation (HHPB) between $[\![P_1]\!]$ and $[\![P_2]\!]$. Figure 5 introduces the variables names and types.

Define $\mathscr{R}_c \subseteq C_1' \times C_2' \times \mathscr{P}(E_1' \times E_2')$ as follows:

$$(y_1, y_2, f_c) \in \mathscr{R}_c \iff \begin{cases} (\pi_1(y_1), \pi_2(y_2), \pi_1 \circ f) \in \mathscr{R} \\ f_c(e) = (\pi_1 \circ f(e)), \pi_2(e)) \in y_2 \text{ for all } e \in y_1 \end{cases}$$

Informally (y_1, y_2, f_c) is in the relation $\mathscr{R}_c$ if there is (x_1, x_2, f) in $\mathscr{R}$ such that x_i is the first projection of y_i and such that f_c satisfies the property: for $(e_1, e_q) \in E_1'$, $f_c(e_1, e_q) = (f(e_1), e_q)$ and $(f(e_1), e_q) \in E_2'$.

Let us show that $\mathscr{R}_c$ is a HHPB between $\langle E_1', C_1', \ell_1' \rangle$ and $\langle E_2', C_2', \ell_2' \rangle$.

- $(\emptyset, \emptyset, \emptyset) \in \mathscr{R}_c$;
- For $(y_1, y_2, f_c) \in \mathscr{R}$ we show that f_c is label and order preserving bijection. We have that f_c is defined as $f_c(e) = (\pi_1 \circ f(e)), \pi_2(e))$, for some f label and order preserving bijection such that $(\pi_1(y_1), \pi_2(y_2), \pi_1 \circ f) \in \mathscr{R}$.

  That f_c is a bijection follows from f a bijection.

  Let $e \in y_1$ with $\pi_1(e) = e_1$, $\pi_2(e) = e_q$, then $f_c(e) = (f(e_1), e_q)$ for some f_c s.t. $(\pi(y_1), \pi_2(y_2), f) \in \mathscr{R}$. We have that $\ell_1'(e) = (\ell_1(e_1), \ell_Q(e_q))$, hence

  $$\ell_2'(f_c(e)) = \ell_2'(f(e_1), e_q) = \big(\ell_2(f(e_1)), \ell_Q(e_q)\big)$$

  As f is label preserving we get $\ell_2'(f_c(e)) = (\ell_1(e_1), \ell_Q(e_q))$, hence $\ell_1'(e) = \ell_2'(f_c(e))$.

  Let us now show that for $e, e' \in y_1$, if $e \leq_{y_1} e'$ then $f_c(e) \leq_{y_2} f_c(e')$. We denote $\pi_1(e) = e_1$, $\pi_2(e) = e_q$ and $\pi_1(e') = e_1'$, $\pi_2(e') = e_q'$. Then from Proposition 3

  $$e \leq_{y_1} e' \implies e_1 \leq_{\pi_1(y_1)} e_1' \text{ or } e_q \leq_{\pi_2(y_1)} e_q' \tag{8}$$

  We consider the case where $e_1 \leq_{\pi_1(y_1)} e_1'$. As f is order preserving we have that $f(e_1) \leq_{\pi_1(y_2)} f(e_1')$. Then $(f(e_1), e_q) \leq_{x_2} (f(e_1'), e_q')$, as the projections are order reflecting.

- Let $(y_1, y_2, f_c) \in \mathscr{R}_c$ and $y_1 \xrightarrow{e''} y_1'$, $y_1' = y_1 \cup \{e''\}$. We consider only the case when $\pi_1(e'') = e_1'' \neq \star$, $\pi_2(e'') = e_q'' \neq \star$ as the rest is similar. From the definition of the projections $\pi_1(y_1)$, $\pi_1(y_1') \in C_1'$ and as $\pi_1(e'') = e_1'' \neq \star$, we have that $\pi_1(y_1') = \pi_1(y_1) \cup \{e_1''\}$. We reason similarly on $\pi_2(y_1)$ and get

  $$\pi_1(y_1) \xrightarrow{e_1''} \pi_1(y_1') \text{ and } \pi_2(y_1) \xrightarrow{e_q''} \pi_2(y_1'). \tag{9}$$

  From Equation 9 and as $(\pi_1(y_1), \pi_2(y_2), f) \in \mathscr{R}$, by definition of $\mathscr{R}_c$, we have that

  $$\exists x_2' \text{ s.t. } \pi_1(y_2) \xrightarrow{e_2''} x_2' = x_2 \cup \{e_2''\} \tag{10}$$

  and

  $$f' = f \cup \{e_1'' \leftrightarrow e_2''\} \tag{11}$$

  such that $(x_1', x_2', f') \in \mathscr{R}$. From Equation 9 and Equation 10 we have that $\exists y_2' \in (\llbracket P_2 \rrbracket \times \llbracket P_Q \rrbracket)$ with $\pi_1(y_2') = x_2'$, $\pi_2(y_2') = \pi_2(y_1')$ and $\exists e_2' \in y_2'$, $\pi_1(e_2') = e_2''$, $\pi_2(e_2') = e_q''$.

  Let us show that $y_2' \notin X_2$. We have that $y_2' \notin X_2$. As $\ell(e_1'')$ and $\ell(e_q'')$ are compatible, then so are $\ell(e_2'')$ and $\ell(e_q'')$, hence $y_2 \cup \{(e_2'', e_q'')\} \notin X_2$.

  Remains to show $(y_1', y_2', f_c') \in \mathscr{R}$, where $f_c' = f_c \cup \{e_1'' \leftrightarrow e_2''\}$. We have that $(\pi_1(y_1'), \pi_1(y_2'), f') \in \mathscr{R}_c$ and from Equation 11 that $\pi_1 \circ f_c' = f'$. □