Theoretical Computer Science 102 (1992) 165-183
Elsevier

Maximality preserving bisimulation

Raymond Devillers*

Laboratoire d'Informatique Théorique, Université Libre de Bruxelles, Boulevard du Triomphe, B-1050 Bruxelles, Belgium

Communicated by M. Nivat
Received March 1990
Revised February 1991

Abstract

Devillers, R., Maximality preserving bisimulation, Theoretical Computer Science 102 (1992) 165-183.

A new bisimulation notion is introduced for the specification of concurrent systems, which resists to a large class of action refinements, even in the presence of invisible actions. The work is presented in the context of labelled P/T nets, but it may be transported to other popular frameworks like prime event structures, process graphs, etc.

1. Introduction

Bisimulation, which is also sometimes called bisimilarity, has been introduced in [9] as a concept which is essentially equivalent to observational equivalence [8]. Its great importance and usefulness for the comparison of different concurrent systems and for proofs of their correctness has been stressed in the literature.

Usually, bisimulation is defined in terms of execution sequences, i.e. in terms of arbitrary interleaving. In this case, however, bisimulation cannot distinguish between a concurrent system and its sequential simulation. Moreover, it does not resist to the simplest action refinements for concurrent systems [11].

In a previous work [2], the author, together with Best, Kiehn and Pomello, defined a fully concurrent bisimulation notion for systems represented by labelled P/T nets with a semantics expressed through the labelled partial orders corresponding to their processes. This relation essentially corresponds to other equivalence notions developed independently in other frameworks, like the BS-bisimulation [13] for behaviour structures and the history preserving bisimulation [5] for event structures. Some nice results have been obtained on action refinements, but some problems

* Research supported by ESPRIT Basic Research Action, project 3148: DEMON.

remained in the presence of invisible actions. This problem has been solved for process graphs by [7] who defined a branching bisimulation, which resists to refinements, and may be transported to the labelled P/T net theory, but only for sequential systems.

We will here introduce a strengthening of the fully concurrent bisimulation, which we called the maximality preserving bisimulation, which still captures the concurrent semantics of labelled systems and withstands a large class of refinements even in the presence of τ-transitions. When applied to sequential systems, our notion proves to be weaker than the branching bisimulation.

2. Basic definitions

2.1. Unlabelled systems

We briefly recall the definitions of some basic concepts, referring e.g. to [3, 1]. A net with arc weights is a triple $N=(S, T, W)$ with $S \cap T = \emptyset$ and

$$W: ((S \times T) \cup (T \times S)) \to \mathbf{N} = \{0, 1, 2, \ldots\}.$$

T is a set of transitions and S is a set of places. We assume all nets to be finite, i.e. $|S \cup T| \in \mathbf{N}$. A net $N=(S, T, W)$ is ordinary iff for all $(x, y) \in ((S \times T) \cup (T \times S))$: $W(x, y) \leq 1$. In an ordinary net, the weight function can (and will) be replaced by a flow relation $F \subseteq ((S \times T) \cup (T \times S))$, following the rule that $(x, y) \in F \Leftrightarrow W(x, y) \neq 0$. For $x \in S \cup T$, the pre-set ${}^\bullet x$ is defined as ${}^\bullet x = \{y \in S \cup T \mid W(y, x) \neq 0\}$ and the post-set $x^\bullet$ is defined as $x^\bullet = \{y \in S \cup T \mid W(x, y) \neq 0\}$.

A marking of a net (S, T, W) is defined as a function $M: S \to \mathbf{N}$, giving the number of tokens contained in each place. The transition rule states that a transition t is enabled by M iff $M(s) \geq W(s, t)$ for all $s \in S$, and that an enabled transition t may occur, producing a successor marking M' by the rule $M'(s) = M(s) - W(s, t) + W(t, s)$ for all $s \in S$. The occurrence of t is denoted by $M[t\rangle M'$.

Two transitions t_1, t_2 (not necessarily distinct) are concurrently enabled by a marking M iff $M(s) \geq W(s, t_1) + W(s, t_2)$ for all $s \in S$. This may be extended to sets or bags of transitions.

A system net (or a marked P/T net) (S, T, W, M_0) is a net (S, T, W) with an initial marking M_0.

A sequence $\sigma = M_0 t_1 M_1 t_2 \ldots$ is an occurrence sequence iff $M_{i-1}[t_i\rangle M_i$ for $1 \leq i$. A sequence $t_1 t_2 \ldots$ is a transition sequence (starting with M) iff there is an occurrence sequence $M t_1 M_1 t_2 \ldots$ If the finite sequence $t_1 t_2 \ldots t_n$ leads from M to M' then we write $M[t_1 t_2 \ldots t_n\rangle M'$. The set of reachable markings of a marked net (S, T, W, M_0) is defined as $[M_0\rangle = \{M \mid \exists t_1 t_2 \ldots t_n: M_0[t_1 t_2 \ldots t_n\rangle M\}$. A marked net (S, T, W, M_0) is safe iff $\forall M \in [M_0\rangle, \forall s \in S: M(s) \leq 1$.

An occurrence net $N=(B, E, F)$ is an acyclic ordinary net without branched places, i.e., $\forall x, y \in B \cup E: (x, y) \in F^+ \Rightarrow (y, x) \notin F^+$ (acyclicity) and $\forall b \in B: |{}^\bullet b| \leq 1 \wedge |b^\bullet| \leq 1$ (no branching of places). For an occurrence net (B, E, F), the pair $(X, \prec)$ with $X = B \cup E$ and $\prec = F^+$ is a strict partial order. Often, elements of E are called events and elements of B are called conditions.

A B-cut $c \subseteq B$ of an occurrence net (B, E, F) is a maximal unordered set of B-elements (taking F^+ as the ordering). $\downarrow c$ denotes the set of elements $\{x \in B \cup E \mid \exists y \in c: (x, y) \in F^*\}$, i.e., the set of elements below or on c. For two B-cuts c_1, c_2 of $N=(B, E, F)$, we define $c_1 \sqsubseteq c_2$ iff $c_1 \subseteq \downarrow c_2$, and $c_1 \sqsubset c_2$ iff $c_1 \sqsubseteq c_2$ and $c_1 \neq c_2$.

In order to avoid minor but annoying technical difficulties, we will suppose from now on that all our nets are T-restricted, i.e., $\forall t \in T: {}^\bullet t \neq \emptyset \neq t^\bullet$.

$Min(N)$ and $Max(N)$ are the B-cuts defined by the sets $\{x \in B \cup E \mid {}^\bullet x = \emptyset\}$ and $\{x \in B \cup E \mid x^\bullet = \emptyset\}$, respectively.

A process $\pi = (N, p) = (B, E, F, p)$ of a system $\Sigma = (S, T, W, M_0)$ consists of an occurrence net $N=(B, E, F)$ together with a labelling $p: B \cup E \to S \cup T$ which satisfy appropriate properties such that π can be interpreted as a concurrent run of Σ, i.e.,

- $p(B) \subseteq S$, $p(E) \subseteq T$ (B-elements are instances of place holdings, E-elements are occurrences of transitions);
- $Min(N)$ is a B-cut which corresponds to the initial marking M_0, that is, $\forall s \in S: M_0(s) = |p^{-1}(s) \cap Min(N)|$;
- $\forall e \in E, \forall s \in S: W(s, p(e)) = |p^{-1}(s) \cap {}^\bullet e|$ and $W(p(e), s) = |p^{-1}(s) \cap e^\bullet|$ (transition environments are respected).[1]

The initial process of Σ is the one for which $E = \emptyset$; it will generally be denoted as π°.

If a marking M of Σ and a B-cut c of a process π of Σ satisfy $\forall s \in S: M(s) = |p^{-1}(s) \cap c|$ then M is said to correspond to c.

If c is a B-cut of a process $\pi = (B, E, F, p)$, then $\Downarrow(\pi, c)$ denotes the process $(B \cap \downarrow c, E \cap \downarrow c, F \cap (\downarrow c \times \downarrow c), p|_{\downarrow c})$, i.e., the prefix of π up to (and including) c.

A process π of a system Σ is an extension of another process π' of the same system if there is a B-cut c of π such that $\pi' = \Downarrow(\pi, c)$; π' is also called a prefix of π.

The set $Lin(\pi)$, for a process π of Σ, defines the set of all occurrence sequences of Σ which are linearizations (of the events and their separating B-cuts) of π. It is known that if Σ is finite then each finite process has a nonempty *Lin*-set. $Lin(\Sigma)$ will denote the set of all the occurrence sequences of Σ.

For an occurrence sequence σ of Σ, $\Pi(\sigma)$ denotes the set of all processes of Σ (up to isomorphism) such that σ linearizes π; Π is thus the inverse of *Lin*. $\Pi(\Sigma)$ will denote the set of all the processes of Σ (up to isomorphism, again).

A system $\Sigma = (S, T, W, M_0)$ is sequential iff $\forall M \in [M_0\rangle$ no two transitions are concurrently enabled; that implies that for any process π of Σ, $|Lin(\pi)| = 1$ and π defines a total order on its events.

[1] For infinite processes there needs to be an additional requirement, but we will not be interested in infinite processes here; the interested reader may find an extensive discussion on this subject in [1].

2.2. Labelled systems

An alphabet A is a finite set of visible actions: we assume that $\tau \notin A$ (τ will denote the internal or silent action).

A labelling of a net $N = (S, T, W)$ is a function $\lambda : T \to A \cup \{\tau\}$. If $\lambda(t) \in A$ then t is called visible; otherwise, t is called silent or invisible.

$\Sigma = (S, T, W, M_0, \lambda)$ is a labelled system, or a labelled P/T net, iff (S, T, W, M_0) is a system net and λ is a labelling of (S, T, W).

An action $a \in A$ in a labelled system $\Sigma = (S, T, W, M_0, \lambda)$ is said to be auto-concurrent at a marking M iff M concurrently enables two observable transitions t_1, t_2 (not necessarily distinct) such that $\lambda(t_1) = \lambda(t_2) = a$. Σ is free of auto-concurrency iff for all $M \in [M_0\rangle$ no observable action is auto-concurrent at M. Absence of auto-concurrency may be viewed as a kind of safeness, it has also been termed the disjoint labelling condition in [12, 11].

An observable transition t of a labelled system $\Sigma = (S, T, W, M_0, \lambda)$ is said to be self-concurrent at a marking M iff M concurrently enables t twice. Σ is free of self-concurrency iff for all $M \in [M_0\rangle$ no observable transition is self-concurrent at M.

Let $\Sigma = (S, T, W, M_0, \lambda)$ be a labelled system. Let $\Pi = (B, E, F, p)$ be a process of it. Then the abstraction of π with respect to λ is denoted by $\alpha_\lambda(\pi) = (E', <, \lambda')$ and is defined by

$$E' = \{e \in E \mid \lambda(p(e)) \neq \tau\},$$

$$< = \{(e_1, e_2) \in E' \times E' \mid (e_1, e_2) \in F^+\},$$

$$\lambda' = \lambda \circ (p|_{E'}), \text{ i.e. } \forall e \in E' : \lambda'(e) = \lambda(p(e)).$$

$(E', <)$ is a partially ordered set, and $\alpha_\lambda(\pi) = (E', <, \lambda')$ is thus a labelled poset (with labels in A).

Let $\alpha_{\lambda_1} = (E'_1, <_1, \lambda'_1)$ and $\alpha_{\lambda_2} = (E'_2, <_2, \lambda'_2)$ be two abstractions as in the previous definition, both with labels in A. Then $\alpha_{\lambda_1} \cong \alpha_{\lambda_2}$ iff there is a bijection $\beta : E'_1 \to E'_2$ such that

(i) $\forall e \in E'_1 : \lambda'_1(e) = \lambda'_2(\beta(e))$,

(ii) $\forall e_1, e_2 \in E'_1 : e_1 <_1 e_2 \Leftrightarrow \beta(e_1) <_2 \beta(e_2)$,

i.e. these abstractions are order-isomorphic.

In the following, we will suppose that all our systems have their labels in the same alphabet A, and we will denote by $A(\Sigma)$ the subset of A which is actually used by a system Σ, i.e. if $\Sigma = (S, T, W, M_0, \lambda)$: $A(\Sigma) = \lambda(T) \setminus \{\tau\}$.

3. Bisimulation and refinement

Intuitively, two systems are bisimilar if there is a bisimulation between them, i.e. a relation between their evolutions such that for each evolution of one of the systems

there is a corresponding evolution of the other system such that the evolutions are observationally "equivalent" and lead to systems which are again bisimilar [10].

If the semantics of labelled system nets is captured by the abstractions of their processes, various bisimulation notions may be introduced (see [2]); for instance the fully concurrent bisimulation (also called history preserving bisimulation or BS-bisimulation in other contexts) may be defined as follows.

Definition 3.1 (*Fully concurrent bisimulation*). If Σ_1 and Σ_2 are two labelled systems, $\Sigma_1 \approx_{FCB} \Sigma_2$ iff there is a set $\mathcal{B} \subseteq \{(\pi_1, \pi_2, \beta) \mid \pi_1 \in \Pi(\Sigma_1), \pi_2 \in \Pi(\Sigma_2), \beta$ is a relation between the visible events of π_1 and $\pi_2\}$ with the following properties:

(i) $(\pi_1^\circ, \pi_2^\circ, \emptyset) \in \mathcal{B}$, where π_1° and π_2° are the initial processes of Σ_1 and Σ_2, respectively.

(ii) $(\pi_1, \pi_2, \beta) \in \mathcal{B} \Rightarrow \beta$ is an order-isomorphism between $\alpha_{\lambda^1}(\pi_1)$ and $\alpha_{\lambda^2}(\pi_2)$

(iii) $\forall (\pi_1, \pi_2, \beta) \in \mathcal{B}$ (a) if π_1' is an extension of π_1, there is $(\pi_1', \pi_2', \beta') \in \mathcal{B}$ where π_2' is an extension of π_2 and $\beta \subseteq \beta'$, (b) vice versa.

Σ_1 and Σ_2 are then said to be FC-bisimilar.

A natural question about an equivalence relation concerns its resistance to some operation or transformation rule, i.e. its congruency. In our context, that means that if two systems are bisimilar and we transform them accordingly, will the transformed systems be again bisimilar? We shall here consider the refinement of all the transitions with a same visible label by some refinement system. As the various transitions to be replaced may have different surrounding shapes, one has first to carefully define how to connect each copy of the refinement system to the neighbourhood of the replaced transitions. This may be done through a multiplicative place interface, where the interconnection is implemented by a matrix of places whose rows correspond to the connection to and from the neighbourhood, whose columns correspond to the connection to and from the refinement system, and where the weights and initial markings are multiples of the original ones in order to homogenize the characteristics. However, this is rather difficult to define formally in the general case (it has been done for systems with arc weights 1 and no side conditions in [6]), but we will prefer here to impose conditions on the refinement systems rather than on the systems to be refined. Consequently, we will restrict ourselves here to simple refinement systems, where the interconnection is easy to define (it is a very simple form of the multiplicative interface) and leads nevertheless to interesting results.

Definition 3.2 (*Empty in/out system*). A labelled system $D = (S^D, T^D, W^D, M_0^D, \lambda^D)$ will be called an empty in/out system iff

(i) there is a unique (input) place s_{in} without predecessor and a (different) unique (output) place s_{out} without successor: $\forall t \in T^D$: $W^D(t, s_{in}) = 0 = W^D(s_{out}, t)$ and $s_{in} \neq s_{out}$,

(ii) initially there is a unique token in s_{in} and at the end there is a unique token in s_{out}: $M_0^D(s_{in}) = 1$ and $\forall s \neq s_{in}$: $M_0^D(s) = 0$; $\forall M \in [M_0^D\rangle$: $M(s_{out}) > 0 \Rightarrow [M(s_{out}) = 1 \wedge \forall s \neq s_{out}$: $M(s) = 0]$,

(iii) s_{in} and s_{out} only have ordinary arcs, $\forall t \in T^D$: $W^D(s_{in}, t) \leq 1 \geq W^D(t, s_{out})$.

We called it an empty in/out system since at the beginning and at the end, there are no marked places between s_{in} and s_{out}.

Definition 3.3 (*Empty refinement*). Let $\Sigma = (S, T, W, M_0, \lambda)$ be a labelled system, let $a \in A(\Sigma)$ and let $D = (S^D, T^D, W^D, M_0^D, \lambda^D)$ be an empty in/out system. The refinement $ref(\Sigma, a, D)$ is the labelled system obtained from Σ by applying the following construction for each $t \in \lambda^{-1}(a)$ (the order does not matter, but the λ used here is the one of Σ and not the one constructed for $ref(\Sigma, a, D)$ below).

(i) Drop t (and restrict W and λ accordingly).

(ii) Create a copy of D without s_{in} and s_{out}; the new nodes will be called $\langle x, t\rangle$ for $x \in S^D \cup T^D$; W, M_0 and λ will be modified accordingly, i.e.

$$\forall x, y \in S^D \cup T^D\text{: } W(\langle x, t\rangle, \langle y, t\rangle) = W^D(x, y),$$

$$\forall x \in T^D\text{: } \lambda(\langle x, t\rangle) = \lambda^D(x)$$

and

$$\forall x \in S^D \backslash \{s_{in}, s_{out}\}\text{: } M_0(\langle x, t\rangle) = 0.$$

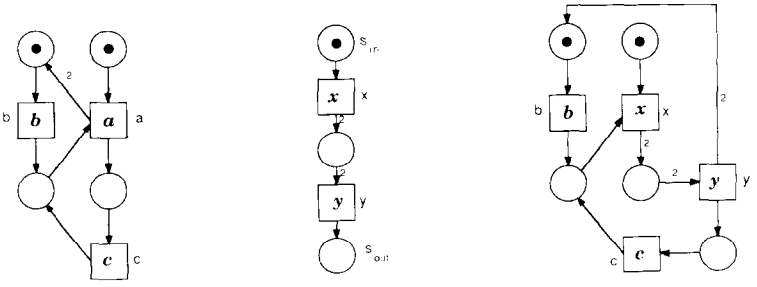
(iii) Connect the successors of s_{in} to the predecessors of t and the predecessors of s_{out} to the successors of t:

$$\forall x \in s_{in}^{\bullet}, \forall y \in {}^{\bullet}t\text{: } W(y, \langle x, t\rangle) = W(y, t),$$

$$\forall x \in {}^{\bullet}s_{out}, \forall y \in t^{\bullet}\text{: } W(\langle x, t\rangle, y) = W(t, y).$$

It may be noticed that, at the end, $\Sigma' = ref(\Sigma, a, D)$ will be a labelled system with $A(\Sigma') = A(\Sigma) \backslash \{a\} \cup A(D)$. Fig. 1 shows an example of this construction.

Definitions 3.2 and 3.3 may seem rather restrictive but



(i) Σ (ii) D (iii) $ref(\Sigma, a, D)$

Fig. 1. Illustration of Definition 3.3.

- they encompass classical subcases like simple splitting, simple choice, renaming (we will address them explicitely later);
- some authors use silent input and output interfaces to connect the refinement system copies, but if we incorporate them in the in/out system itself we are exactly in the conditions described in 3.2 and 3.3;
- the theory may be extended to in/out systems with an internal marking and a memoryless constraint, like in [14], but this renders the treatment still more intricate and we will not do it here;
- even so, rather awkward situations may occur, as exhibited in Fig. 2.

A careful examination of the example in Fig. 2 shows that the problem originates from the presence of a silent transition immediately after an a-labelled one; this led in [2] to some congruence results on FC-bisimulation, under constraints excluding the mentioned "bad" configurations. The problem may also be solved by strengthening the bisimulation definition in order to distinguish systems like the ones exhibited in Fig. 2. This has been done in [7] for sequential systems in the context of process graphs.

For a sequential system, the labelled posets of the processes are labelled total orders and, translating [7] in our system net framework, we may define the branching bisimulation as follows.

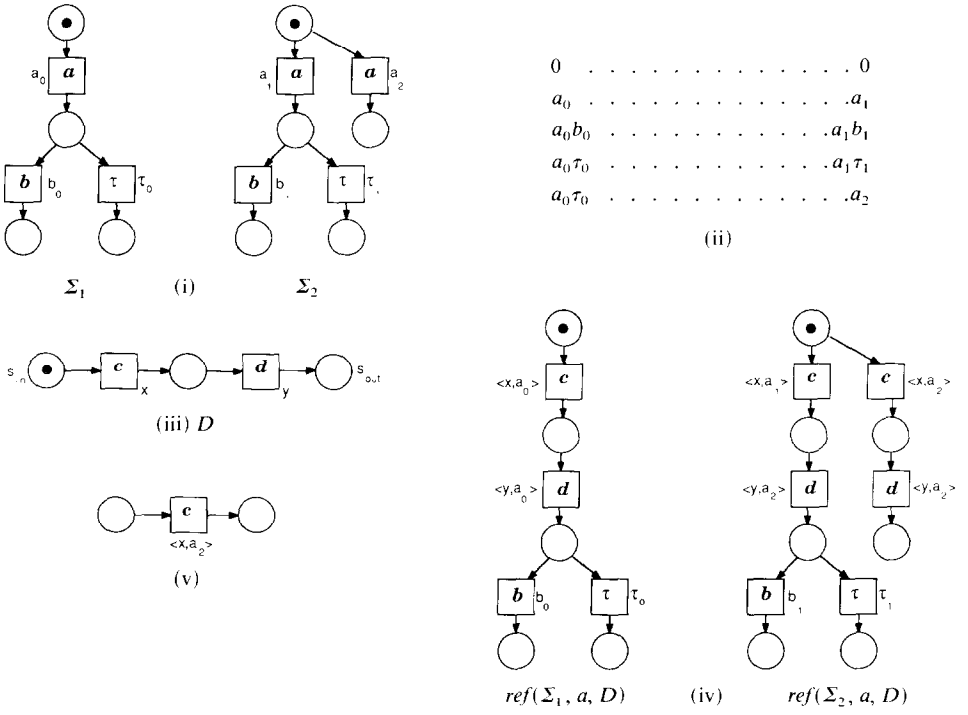


Fig. 2. (i) Two FC-bisimilar labelled systems. (ii) The corresponding processes. (iii) An empty in/out system. (iv) The refinements of Σ_1 and Σ_2. (v) A process of $ref(\Sigma_2, a, D)$ whose extensions are not the same as the ones for the only possible corresponding process of $ref(\Sigma_1, a, D)$.

Definition 3.4 (*Branching bisimulation*). If Σ_1 and Σ_2 are two sequential labelled systems, $\Sigma_1 \approx_{brB} \Sigma_2$ iff there is a relation $\rho \subseteq \Pi_1 \times \Pi_2$ such that

(i) $(\pi_1^\circ, \pi_2^\circ) \in \rho$, where π_1° and π_2° are the initial processes of Σ_1 and Σ_2, respectively,

(ii) $(\pi_1, \pi_2) \in \rho \Rightarrow \alpha_{\lambda^1}(\pi_1) \cong \alpha_{\lambda^2}(\pi_2)$ (they are order-isomorphic),

(iii) $\forall (\pi_1, \pi_2) \in \rho$ (a) if π_1' is an extension of π_1 with only one event more, there is $(\pi_1', \pi_2') \in \rho$ such that π_2' is an extension of π_2 and any process π_2'' strictly between π_2 and π_2' (i.e. π_2 is a strict prefix of π_2'' and π_2'' is a strict prefix of π_2') satisfies $(\pi_1, \pi_2'') \in \rho$, (b) vice versa.

Σ_1 and Σ_2 will then be said br-bisimilar.

This equivalence notion presents a nice "sandwich" property besides the usual "extension" one. Moreover, the translation of the results of [7] in terms of labelled system nets shows that the branching bisimulation withstands empty refinements for sequential systems. For instance, the sequential systems Σ_1 and Σ_2 in Fig. 2 are not br-bisimilar, as exhibited by the last (necessary) correspondence between their processes.

4. Maximality preserving bisimulation

We shall now address the problem to define a bisimulation notion for labelled concurrent systems, which withstands empty refinements, maybe under some reasonable constraints. Since we want to refine some visible action(s), we may introduce them explicitly in the notion.

A careful examination of the difficulties encountered in [2] leads to the following definition.

Definition 4.1 (*Bisimulation preserving maximality for a visible action set*). If $\Sigma_1 = (S_1, T_1, W_1, \lambda_1)$ and $\Sigma_2 = (S_2, T_2, W_2, \lambda_2)$ are two labelled systems and $\mathscr{A} \subseteq A$ is a set of visible actions, $\Sigma_1 \approx_{\mathscr{A}MPB} \Sigma_2$ iff there is a set $\mathscr{B} \subseteq \{(\pi_1, \pi_2, \beta) \mid \pi_1 \in \Pi(\Sigma_1), \pi_2 \in \Pi(\Sigma_2), \beta$ is a relation between the visible events of π_1 and $\pi_2\}$ with the following properties:

(i) $(\pi_1^\circ, \pi_2^\circ, \emptyset) \in \mathscr{B}$, where π_1° and π_2° are the initial processes of Σ_1 and Σ_2, respectively,

(ii) $(\pi_1, \pi_2, \beta) \in \mathscr{B} \Rightarrow \beta$ is an order-isomorphism between $\alpha_{\lambda^1}(\pi_1)$ and $\alpha_{\lambda^2}(\pi_2)$,

(iii) $\forall (\pi_1, \pi_2, \beta) \in \mathscr{B}$, with $\pi_1 = (B_1, E_1, F_1, p_1)$ and $\pi_2 = (B_2, E_2, F_2, p_2)$

(a) if $\pi_1' = (B_1', E_1', F_1', p_1')$ is an extension of π_1 with only one event e_1' more, there is $(\pi_1', \pi_2', \beta') \in \mathscr{B}$ where $\pi_2' = (B_2', E_2', F_2', p_2')$ is an extension of π_2 and $\beta \subseteq \beta'$; moreover,

- if $\lambda_1(p_1'(e_1')) \in \mathscr{A}$, then $\beta'(e_1')$ is a maximal event of π_2'
- for any $e_1 \in E_1$, if $\lambda_1(p_1(e_1)) \in \mathscr{A}$ and if e_1 and $\beta(e_1)$ are maximal events in π_1' and π_2, respectively, then $\beta(e_1)$ is still a maximal event in π_2',

(b) vice versa.

Σ_1 and Σ_2 will then be said $\mathscr{A}$MP-bisimilar.

The intuitive meaning of this definition is that the corresponding processes have to be order-isomorphic, that the initial processes correspond to each other and that they present an "extension property" (any extension of a process corresponds to an extension of any corresponding process; in the definition one only considers extensions by a single event on one side, but one may iterate to get any extension); the last conditions essentially say that the maximality of $\mathscr{A}$-labelled events may be preserved: if a new $\mathscr{A}$-labelled event is added on one side (it is then maximal), it is possible to extend the other side in such a way that the corresponding event (with the same label) is also maximal, and if an $\mathscr{A}$-labelled event is maximal on one side before and after the extension while on the other side the corresponding event is also maximal before the extension, then this event remains maximal after the extension too.

It may be observed that Definition 4.1 is essentially the same as Definition 3.1 up to the additional constraints on $\mathscr{A}$-labelled events; more precisely, we have the following corollary.

Corollary 4.2 *FC-bisimulation is a maximality preserving bisimulation*:

$$\approx_{FCB} = \approx_{\emptyset MPB}.$$

Proof. Obvious. □

Definition 4.3 (*Maximality Preserving Bisimulation*). Two systems will be said MP-bisimilar if they are maximality preserving bisimilar for all visible actions, i.e. $\approx_{MPB} = \approx_{AMPB}$.

If $\mathscr{A} = \{a\}$, we will simply write $\approx_{aMPB}$ instead of $\approx_{\{a\}MPB}$, and say that two systems are maximality preserving bisimilar with respect to a, instead of $\{a\}$.

Corollary 4.4 (Strengthening property). *If* $\mathscr{A}' \subset \mathscr{A} \subseteq A$, *then* $\approx_{\mathscr{A}MPB}$ *is stronger than* $\approx_{\mathscr{A}'MPB}$; *and in particular MP-bisimulation is stronger than FC-bisimulation.*

Proof. It is clear that $\approx_{\mathscr{A}MPB}$ is stronger or identical to $\approx_{\mathscr{A}'MPB}$ since the additional conditions concern more actions in $\mathscr{A}$MP-bisimulation. The fact that the strengthening is strict results from the observation that the two systems Σ_1 and Σ_2 in Fig. 2 are not aMP-bisimilar (since the addition of a_2 to the initial process on the right size needs the addition of a_0 and τ_0 on the left size, where a_0 is not maximal), while they are FC-bisimilar. The same example where a is replaced by any action in $\mathscr{A} \setminus \mathscr{A}'$ will thus exhibit the strict strengthening property. □

Corollary 4.5 (Widening property). $\forall \Sigma_1, \Sigma_2 : \Sigma_1 \approx_{\mathscr{A}MPB} \Sigma_2 \Rightarrow \Sigma_1 \approx_{\mathscr{A}'MPB} \Sigma_2$ *with* $\mathscr{A}' = \mathscr{A} \cup [A \setminus (A(\Sigma_1) \cap A(\Sigma_2))]$.

Proof. Obvious; this simply means that we may always add to $\mathscr{A}$ any event which never actually occurs in Σ_1 and Σ_2; notice that, normally, if two labelled systems

are bisimilar they have the same alphabet, but it could happen that all the transitions corresponding to some action are dead; consequently, it may happen that $A(\Sigma_1) \neq A(\Sigma_2)$, but then only the actions in $A(\Sigma_1) \cap A(\Sigma_2)$ may occur (and still not necessarily all of them); one could say that the actions in $(A(\Sigma_1) \setminus A(\Sigma_2)) \cup (A(\Sigma_2) \setminus A(\Sigma_1))$ only occur syntactically in Σ_1 and Σ_2, and not semantically. □

Even so, rather awkward situations may still occur, as exhibited in Fig. 3. A careful examination of the example in Fig. 3 shows that the problem here arises from the combination of the fact that a refined transition is self-concurrent and that in the refinement system there is a transition needing more than one token.

There are thus two ways to overcome the difficulty: either by excluding self-concurrency or by excluding multiple needs.

Before attacking these two points, let us first develop some preliminary remarks, which will ease the proof of congruence properties by giving a general framework for these proofs (and which also establishes links with other frameworks).

First, we may define directly refinements on the processes.

Definition 4.6 (*Refinement of a process*). Let $\pi = (B, E, F, p)$ be a process of a labelled system $\Sigma = (S, T, W, M_0, \lambda)$; let $a \in A(\Sigma)$ be a visible action and let D be an empty in/out system; let $\zeta : p^{-1}(\lambda^{-1}(a)) \to \Pi(D)$ be a function such that if

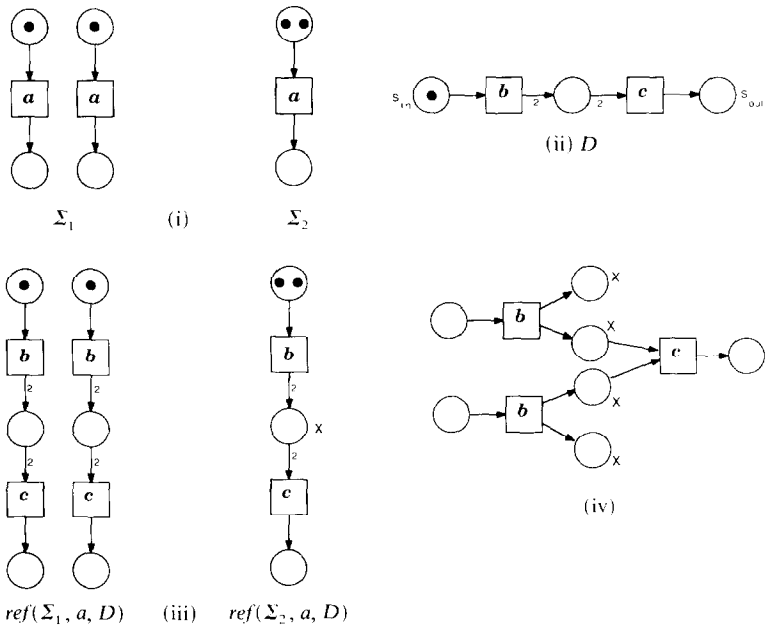


Fig. 3. MP-bisimulation does not always withstand refinements. (i) Two MP-bisimilar labelled systems. (ii) An empty in/out system. (iii) The refinements of Σ_1 and Σ_2. (iv) A process of $ref(\Sigma_2, a, D)$ which has no corresponding process in $ref(\Sigma_1, a, D)$.

$e \in p^{-1}(\lambda^{-1}(a))$ is not a maximal event in π, then $\zeta(e)$ is a complete process of D, i.e. with a (unique) maximal condition corresponding to s_{out}; we will also suppose that $\zeta(e)$ is never the initial process of D. Then the refinement $ref(\pi, a, \zeta)$ is obtained from π by applying the following construction for each $e \in p^{-1}(\lambda^{-1}(a))$ (the order does not matter):

(i) Drop e (and modify F, p accordingly).

(ii) If $\zeta(e)$ is a complete process of D, create a copy of $\zeta(e)$, drop the *Min* and *Max* of it (corresponding to s_{in} and s_{out}); replace the labelling $p^{\zeta(e)}$ of this copy by $p^e : x \to \langle p^{\zeta(e)}(x), p(e) \rangle$, and connect the (unique) minimal event of the copy to the predecessor conditions of e and the (unique) maximal event of the copy to the successor conditions of e.

(iii) If $\zeta(e)$ is not complete, create a copy of $\zeta(e)$, drop the *Min* of it (corresponding to s_{in}) and the successor conditions of e; replace the labelling $p^{\zeta(e)}$ of this copy by $p^e : x \to \langle p^{\zeta(e)}(x), p(e) \rangle$, and connect the (unique) minimal event of the copy to the predecessor conditions of e.

Proposition 4.7. *Refined processes are processes of the refined system. With the notations of the previous definition,* $ref(\pi, a, \zeta) \in \Pi(ref.(\Sigma, a, D))$.

Proof. Trivial but slightly tedious. Notice however that the property is directly connected to the fact that there is no "internal marking" in an empty in/out system, as is exhibited by Fig. 4. □

The example in Fig. 3 shows that the reverse is not true in general: the process in (iv) is not a refinement of a process of Σ_2. But this suggests to define a class of refinements based on it.

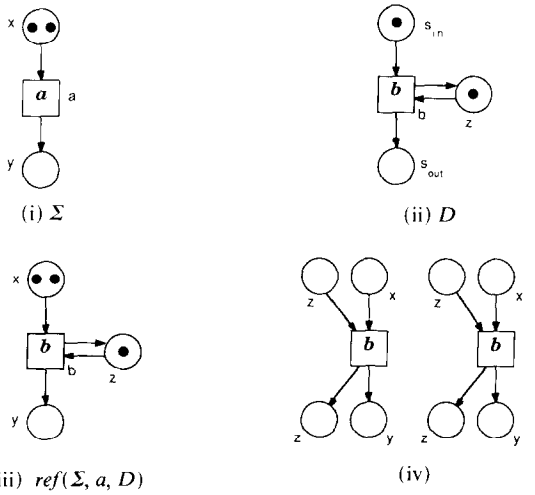


Fig. 4. A nonempty refinement. (i) A labelled system. (ii) A nonempty in/out system. (iii) The refinement of Σ. (iv) A refined process of Σ which is not a process of $ref(\Sigma, a, D)$.

Definition 4.8 (*Refinements with refined processes*). Let Σ be a labelled system, $a \in A(\Sigma)$ and let D be an empty in/out system. We will say that $ref(\Sigma, a, D)$ has refined processes iff $\forall \tilde{\pi} \in \Pi(ref(\Sigma, a, D)), \exists \pi \in \Pi(\Sigma), \exists \zeta$ such that $\tilde{\pi} = ref(\pi, a, \zeta)$ (up to isomorphism). □

Now, our central result is the following.

Theorem 4.9 (Refinements with refined processes respect $\mathscr{A}$MP-bisimulation). *If Σ_1 and Σ_2 are two labelled systems, $a \in \mathscr{A} \subseteq A$ and D is an empty in/out system such that $ref(\Sigma_1, a, D)$ and $ref(\Sigma_2, a, D)$ have refined processes, then*

$$\Sigma_1 \approx_{\mathscr{A}MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{\mathscr{A}'MPB} ref(\Sigma_2, a, D)$$

with $\mathscr{A}' = A \backslash [A(\Sigma_1) \cap A(\Sigma_2)] \cup \mathscr{A}$.

Proof. Let $\mathscr{B}$ satisfy the conditions of Definition 4.1 for Σ_1 and Σ_2, and let us define $\tilde{\mathscr{B}}$ as the set of triples $(\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\beta})$ such that there is a triple $(\pi_1, \pi_2, \beta) \in \mathscr{B}$ and ζ, ζ' with the following properties:

(i) $\tilde{\pi}_1 = ref(\pi_1, a, \zeta)$, $\tilde{\pi}_2 = ref(\pi_2, a, \zeta')$,

(ii) $\zeta' = \zeta \circ \beta^{-1}$, i.e. ζ' is the image of ζ through β,

(iii) $\tilde{\beta}$ is identical to β on the visible events common to $\tilde{\pi}_1$ and π_1 on one side and to $\tilde{\pi}_2$ and π_2 on the other side, and it is the identity relation (restricted to visible events) for the corresponding identical copies (due to the definition of ζ') of the processes of D refining the corresponding a-labelled events of π_1 and π_2.

It should be clear that

- $(\tilde{\pi}_1^\circ, \tilde{\pi}_2^\circ, \emptyset) \in \tilde{\mathscr{B}}$, if $\tilde{\pi}_1^\circ$ and $\tilde{\pi}_2^\circ$ are the initial processes of $ref(\Sigma_1, a, D)$ and $ref(\Sigma_2, a, D)$,
- if $(\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\beta}) \in \tilde{\mathscr{B}}$ then $\tilde{\beta}$ is an order-isomorphism between the abstractions of $\tilde{\pi}_1$ and $\tilde{\pi}_2$, since it is constructed from β which is itself an order-isomorphism and from identity relations,
- if $(\tilde{\pi}_1, \tilde{\pi}_2, \tilde{\beta}) \in \tilde{\mathscr{B}}$ and $\tilde{\pi}_1$ is extended into $\tilde{\pi}_1'$ by an event e_1', three cases are possible.

(a) e_1' extends a D-process refining some a-labelled event e_1 in π_1 and the same prolongation may be applied to the identical process refining $\beta(e_1)$ in π_2; $\tilde{\beta}$ may be extended accordingly and it should be clear that we will obtain an extension triple which is still in $\tilde{\mathscr{B}}$ and which preserves the maximality for all the visible actions.

(b) e_1' does not belong to any D-process refining some a-labelled event in π_1; then e_1' also extends π_1 into π_1' and $\tilde{\pi}_1' = ref(\pi_1', a, \zeta)$; from $\mathscr{A}$MP-bisimulation, there are π_2' and β' extending π_2 and β such that $(\pi_1', \pi_2', \beta') \in \tilde{\mathscr{B}}$ and maximality of $\mathscr{A}$-labelled events is preserved. It should be clear that $\tilde{\pi}_1' = ref(\pi_1', a, \zeta)$, $\tilde{\pi}_2' = ref(\pi_2', a, \zeta)$ and the $\tilde{\beta}'$ corresponding to β' will give a triple belonging to $\mathscr{B}$ with the good properties; indeed, the only possible problem is that an additional event in π_2' would have to be connected in $\tilde{\pi}_2'$ to an incomplete refining process of D (which would be impossible), but then there is the same incomplete process in $\tilde{\pi}_1'$

and they both correspond to maximal β-corresponding a-labelled events e_1'' and e_2'' in π_1' and π_2' respectively; in π_1', e_1 may not be connected to e_1'' (otherwise the corresponding D-process of the latter would be complete), thus e_1'' remains maximal but then so must be e_2'', hence the contradiction. The only visible events the maximality of which could be destroyed are clearly the ones with that property in Σ_1 and Σ_2.

(c) e_1' is the beginning of a new D-process refining some new a-labelled event $\ddot{e}_1$ extending π_1 into π_1'. By extending ζ into ζ' with the pair ($\ddot{e}_1$, D-process corresponding to e_1'), we have $\tilde{\pi}_1' = ref(\pi_1', a, \zeta')$; from $\mathscr{A}$MP-bisimulation, there are π_2' and β' extending π_2 and β such that $(\pi_1', \pi_2', \beta') \in \tilde{\mathscr{B}}$ and maximality of $\mathscr{A}$-labelled events is preserved; it should be clear that $ref(\pi_1', a, \zeta')$, $ref(\pi_2', a, \zeta')$ and the $\tilde{\beta}'$ corresponding to β' will give a triple belonging to $\tilde{\mathscr{B}}$ with the good properties; the same reasoning as in (b) may be resumed, with the additional remark that $\beta'(\ddot{e}_1)$ being also maximal, there is no problem in refining it.

and symmetrically for the vice versa part. Consequently the maximality is preserved for any visible action but the ones for which this is not the case in Σ_1 and Σ_2, hence the formula for $\mathscr{A}'$. □

Corollary 4.10 (Refinements with refined processes respect MP-bisimulation). *If Σ_1 and Σ_2 are two labelled systems, $a \in A$ and D is an empty in/out system such that $ref(\Sigma_1, a, D)$ and $ref(\Sigma_2, a, D)$ have refined processes, then*

$$\Sigma_1 \approx_{MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{MPB} ref(\Sigma_2, a, D)$$

Proof. Immediate from Theorem 4.9. In this case $\mathscr{A} = A = \mathscr{A}'$. □

It may be observed that these results may be transported to other models of concurrency based on partial orders if the behaviours of a refined system look like the refinements of the behaviours, as it is the case for instance in the event structure-based theory developed in [5].

Now, we simply have to determine in what circumstances the refinements of a system net have refined processes.

Following the ideas mentioned while analysing the example of Fig. 3, let us define the following.

Definition 4.11 (*SM-systems and SM-refinements*). An SM-system is an empty in/out system D such that $\forall t \in T^D: |{}^\bullet t| = 1 = |t^\bullet|$ and $W^D({}^\bullet t, t) = 1 = W^D(t, t^\bullet)$, i.e. D is essentially a state machine net.

An SM-refinement is a refinement through an SM-system.

Proposition 4.12 (SM-refinements have refined processes). *Let $\Sigma = (S, T, W, M_0, \lambda)$ be a labelled system, let $D = (S^D, T^D, W^D, M_0^D, \lambda^D)$ be an SM-system and let $a \in A(\Sigma)$; then $ref(\Sigma, a, D)$ has refined processes.*

Proof. The basic observation here is the fact that for any SM-system, all the processes are simple chains. Let $\tilde{\pi}$ be any process of $ref(\Sigma, a, D)$. Any node of $\tilde{\pi}$ whose label does not belong to $S \cup T$

- has a label of the form $\langle x, t\rangle$ for some $t \in \lambda^{-1}(a)$;
- has a unique maximal predecessor e with a label of the form $\langle y, t\rangle$ where y is an immediate successor of s_{in} in D;
- belongs to the unique maximal chain γ originating from e where all the labels are of the form $\langle z, t\rangle$, where $z \in T^D \cup S^D \backslash \{s_{in}, s_{out}\}$.

For any such chain γ, the only connections with the rest of the process are

- through the input condition(s) of e (always);
- through the output condition(s) of the maximal event in the chain if it has a label $\langle u, t\rangle$ where u is an immediate predecessor of s_{out} in D (i.e. if the chain does not stop on a maximal condition before).

If we replace this chain γ by an event $\tilde{e}$ with label t (plus adequate successor conditions if the chain ended on a condition) and if we resume the construction until all the nodes have their labels in $S \cup T$, it should be clear that the resulting object is a process π of Σ, and that if ζ is the function associating to each constructed $\tilde{e}$ the process of D corresponding to γ (by adding an input condition corresponding to s_{in}, and an output condition corresponding to s_{out} if the chain ended on an event), $ref(\pi, a, \zeta) = \tilde{\pi}$. This terminates the proof. □

As immediate corollaries, we have the following.

Corollary 4.13 (SM-refinements preserve MP-bisimulation). *If Σ_1 and Σ_2 are labelled systems, $a \in \mathcal{A} \subseteq A$ and D is an SM-system,*

(a) $\Sigma_1 \approx_{\mathcal{A}MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{\mathcal{A}'MPB} ref(\Sigma_2, a, D)$ *with*

$$\mathcal{A}' = A \backslash [A(\Sigma_1) \cap A(\Sigma_2)] \cup \mathcal{A},$$

(b) $\Sigma_1 \approx_{MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{MPB} ref(\Sigma_2, a, D)$,

(c) $\Sigma_1 \approx_{aMPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{FCB} ref(\Sigma_2, a, D)$.

Proof. Immediate from 4.9, 4.10, 4.4 and 4.12. □

Corollary 4.14 (Renaming, simple splitting and simple choice replacements). *MP-bisimulation is preserved by renaming, simple splitting and simple choice replacements.*

Proof. This results from the fact that the three systems depicted in Fig. 5. are SM-refinement systems. It may be observed that it is not necessary that x and y are distinct, that they are different from other labels already in the bisimilar systems, or that they are visible. □

By attacking the other reason of the failure in Fig. 3, we obtain Proposition 4.15.

Proposition 4.15 (Self-concurrency freeness and refined processes). *Let $\Sigma = (S, T, W, M_0, \lambda)$ be a labelled system, let $D = (S^D, T^D, W^D, M_0^D, \lambda^D)$ be an empty*

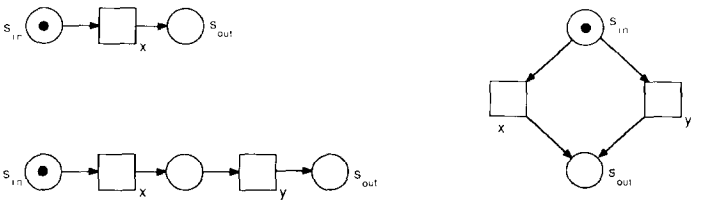


Fig. 5.

in/out system and let $a \in A(\Sigma)$ be a label such that no a-labelled transition is self-concurrent in Σ, then ref(Σ, a, D) has refined processes.

Proof. Let $\tilde{\pi}$ be any process of $ref(\Sigma, aD)$. Any node of $\tilde{\pi}$ whose label does not belong to $S \cup T$ (if any) has a label of the form $\langle x, t\rangle$ for some $t \in \lambda^{-1}(a)$ and has a predecessor with a label $\langle y, t\rangle$ where y is an immediate successor of s_{in} in D.

Let e be a minimal node whose label does not belong to $S \cup T$ (if there is no such label, the construction stops); e is an event with a label $\langle y, t\rangle$ where $y \in s_{in}^{\bullet}$ and $t \in \lambda^{-1}(a)$; moreover, there is no other event with the same property and the same t since the set of nodes without predecessors with a label out of $S \cup T$ is a process of Σ, ${}^{\bullet}e$ is in the *Max* of this process and if there were two such e's corresponding to the same t, from known properties on processes (see [1]), transition t would be self-concurrent in Σ. Consequently, e is a predecessor for all the nodes with a label of the form $\langle x, t\rangle$. Let γ be the maximal structure issued from e, where all the nodes have a label $\langle z, t\rangle$. e is the unique minimal element amongst them.

Now, two cases are possible.

(a) γ does not contain any event with a label $\langle u, t\rangle$ where $u \in {}^{\bullet}s_{out}$. Then, up to the initial condition, γ is a process of D. Indeed, the only problem would be that, at some point, an event in γ has some of its input conditions out of γ, but any of them has a label of the form $\langle v, t\rangle$, is a successor of e and there should be a path from e to that condtion, leaving γ at some point. This could only be through an event with a label $\langle w, t\rangle$ where $w \in {}^{\bullet}s_{out}$, but we supposed there were no such events. Consequently, γ is an (incomplete) process of D and it is completely isolated from its surrounding. We may then replace γ by a new event $e(\gamma)$, with a label t, and the adequate output conditions. $e(\gamma)$ is then a maximal event and the construction may resume, with another (possibly the same) t.

(b) γ contains one or more events e' with a label $\langle u, t\rangle$ where $u \in {}^{\bullet}s_{out}$. If γ has no event with some input conditions out of γ, then, up to the initial condition and the terminal ones corresponding to the various e''s, γ is a process of D. But then, from the definition of an empty refinement, when u is reached there are no tokens left in D, so that e' is unique and there are no "free" conditions (without outgoing arc) in γ. Consequently, γ is definitely isolated from its surrounding, its only connections, present and future, are through the unique input event e and the unique output event e'; we may then replace γ by a new event $e(\gamma)$, with a label t, since the inputs and outputs of γ correspond to those of t, and resume the construction

since again the adequate conditions are satisfied. Now, if there is an event in γ with input conditions out of γ, let c be a minimal condition of this type. c has a label $\langle v, t\rangle$ and is a successor of e. There is thus a path from e to c leaving γ at some point. This may only be at some e'. Since c is minimal, no event between e and e' may have inputs out of γ but then, between e and e', we have a complete process γ' of D and again as there are no tokens left at the end in D, $\gamma = \gamma'$ and we find a contradiction.

Consequently, at the end, we will get a process π of Σ, and if ζ is the function associating with each constructed $e(\gamma)$ the process of D corresponding to γ (by adding an input condition corresponding to s_{in}, and in the second case an output condition corresponding to s_{out}), $ref(\pi, a, \zeta) = \tilde{\pi}$.

This terminates the proof. □

Corollary 4.16 (MP-bisimulation is preserved for systems without self-concurrency). *Let $a \in \mathscr{A} \subseteq A$, let Σ_1 and Σ_2 be two labelled systems such that no a-labelled transition is self-concurrent in them, and let D be an empty in/out system, then*

(a) $\Sigma_1 \approx_{\mathscr{A}MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{\mathscr{A}'MPB} ref(\Sigma_2, a, D)$ *with*

$$\mathscr{A}' = A \backslash [A(\Sigma_1) \cap A(\Sigma_2)] \cup \mathscr{A},$$

(b) $\Sigma_1 \approx_{MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{MPB} ref(\Sigma_2, a, D)$,

(c) $\Sigma_1 \approx_{aMPB} \Sigma_2 \Rightarrow ref(\Sigma_1, a, D) \approx_{FCB} ref(\Sigma_2, a, D)$.

Proof. Immediate from 4.9, 4.10, 4.4 and 4.15. □

Concerning the application domain of Corollary 4.16, let us notice that a 1-safe system net is automatically self-concurrency free.

5. Simultaneous refinements

One may also define refinements simultaneously for a set of visible actions.

Definition 5.1 (*Empty simultaneous refinements*). Let $\Sigma = (S, T, W, M_0, \lambda)$ be a labelled system, let $\mathscr{R} \subseteq A$ be a set of visible actions, and let D be a function: $a \in \mathscr{R} \to D_a$ empty in/out system, associating an in/out system with each action in $\mathscr{R}$. The refinement $ref(\Sigma, \mathscr{R}, D)$ is the labelled system obtained from Σ by applying the construction 3.3(i) to (iii) for each $a \in \mathscr{R}$ and each $t \in \lambda^{-1}(a)$.

Clearly, $ref(\Sigma, a, D_a) = ref(\Sigma, \{a\}, D)$, but $ref(\Sigma, \mathscr{R}, D)$ may not always be obtained by an iterative use of single action refinements since, if $\mathscr{R} = \{a, b, \ldots\}$, it may happen that new b-labelled transitions are added in $ref(\Sigma, a, D_a)$ through D_a.

However, the same result may be obtained by using successive single action refinements through empty in/out systems with disjoined actual alphabets, and then

by applying some renaming, since we have seen in Corollary 4.14 that renamings are a special case of SM-refinements for which there are no problems. Consequently we have the following.

Corollary 5.2 (MP-bisimulation is preserved by simultaneous refinements). *If Σ_1 and Σ_2 are two labelled systems, $\mathcal{R} \subseteq \mathcal{A} \subseteq A$ and D is a function applying $\mathcal{R}$ to a family of empty in/out systems, then if for any $a \in \mathcal{R}$ either $D(a)$ is an SM-system or no a-labelled transition in Σ_1 or Σ_2 is self-concurrent*

(a) $\Sigma_1 \approx_{\mathcal{A}MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, \mathcal{R}, D) \approx_{\mathcal{A}'MPB} ref(\Sigma_2, \mathcal{R}, D)$ *with*

$$\mathcal{A}' = A \backslash [A(\Sigma_1) \cap A(\Sigma_2)] \cup \mathcal{A}$$

(b) $\Sigma_1 \approx_{MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, \mathcal{R}, D) \approx_{MPB} ref(\Sigma_2, \mathcal{R}, D)$

(c) $\Sigma_1 \approx_{\mathcal{R}MPB} \Sigma_2 \Rightarrow ref(\Sigma_1, \mathcal{R}, D) \approx_{FCB} ref(\Sigma_2, \mathcal{R}, D)$.

Proof. Immediate from the above remark and the previous results. □

6. Sequential systems

A natural question now is how the MP-bisimulation compares, for sequential systems, to the branching bisimulation, which is also preserved by refinements in that case (let us notice that all sequential systems are trivially self-concurrency free). The answer is presented now.

Proposition 6.1 (MP-bisimulation is weaker than branching bisimulation). *If Σ_1 and Σ_2 are sequential systems, $\Sigma_1 \approx_{brB} \Sigma_2 \Rightarrow \Sigma_1 \approx_{MPB} \Sigma_2$ but the reverse implication does not hold.*

Proof. Clearly, if $\Sigma_1 \approx_{brB} \Sigma_2$, the relation $\rho \subseteq \Pi_1 \times \Pi_2$ satisfying the criteria 3.4 for the branching bisimulation, together with the trivial isomorphism β between the equally ranked visible events, gives a triple set $\{(\pi_1, \pi_2, \beta)\}$ satisfying the criteria for MP-bisimulation since the only difference is that in addition the intermediate processes correspond to the original process on the other side.

To see that the reverse implication does not hold, one simply has to consider the counterexample shown in Fig. 6.

One can check that $\Sigma_1 \approx_{MPB} \Sigma_2$, but the process π_2 of Σ_2, which is a one event extension of the initial process, may only correspond to the process π_1 of Σ_1, and the intermediate process with only t does not correspond to the initial process of Σ_2. □

7. Conclusion

So far, we have shown that our MP-bisimulation, in the frame of labelled P/T nets, is preserved by SM-refinements, and by empty refinements if we exclude

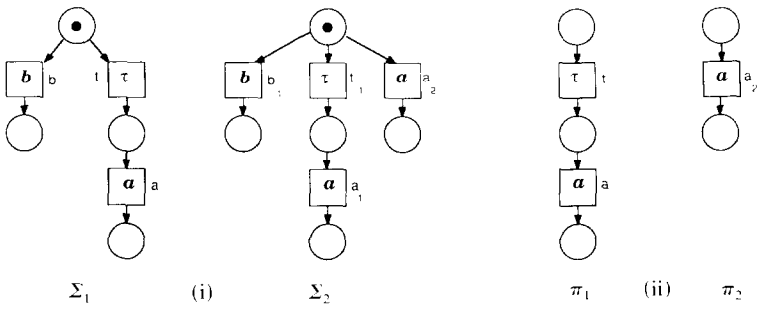


Fig. 6. Two MP-bisimilar systems which are not br-similar. (i) Two MP-bisimilar labelled systems. (ii) Two corresponding processes.

self-concurrency. Our equivalence notion is weaker than branching bisimulation and it encompasses FC-bisimulation. Moreover, we have indicated how to export our results to other behavioural models for concurrency.

And indeed, meanwhile but independently, various authors applied the very same idea to various contexts. For instance, Vogler defined in [15] various refinement congruences for prime event structures and it occurs that his hST-bisimulation essentially corresponds to our MP-bisimulation; moreover, he applied the same idea to interleaving bisimulations, and showed coarsest results.

Similarly, Cherief and Schnoebelen showed in [4] that for process graphs, which may be used to model sequential systems, the Δ-bisimulation defined by a maximality preserving property is preserved by refinements, and is the largest congruence bisimulation.

References

[1] E. Best and R. Devillers, Sequential and concurrent behaviour in Petri net theory, *Theoret. Comput. Sci.* **55** (1) (1988).

[2] E. Best, R. Devillers, A. Kiehn and L. Pomello, Concurrent bisimulations in Petri Nets, *Acta Inform.* **28** (1991) 231-264.

[3] E. Best and C. Fernández, Notation and terminology on Petri net theory, *Arbeitspapiere GMD* **195** (1987).

[4] F. Cherief and P. Schnoebelen, τ-Bisimulations and full abstraction for refinement of actions, Technical Report LIFIA-Imag (Grenoble, France).

[5] R. van Glabbeek and U. Goltz, Equivalence notions for concurrent systems and refinement of actions, *Arbeitspapiere GMD* **366** (1989). Extended abstract in: *Proc. MFCS'89*, Lecture Notes in Computer Science, Vol. 379 (Springer, Berlin, 1989) 237-248.

[6] R. van Glabbeek and U. Goltz, Refinement of actions in causality based models, *Arbeitspapiere GMD* **428** (1990), also in: *Proc. REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness*, Lecture Notes in Computer Science, Vol 430 (Springer, Berlin, 1990) 267-300.

[7] R. van Glabbeek and W. Weijland, Refinement in branching time semantics, in: *Proc. Internat. Conf. on Algebraic Methodology and Software Technology*, Iowa City, USA (1989).

[8] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92 (Springer, Berlin, 1980).

[9] D. Park, Concurrency and automata on infinite sequences, in: P. Deussen, ed., *Proc. 5th GI Conf. on Theoretical Computer Science*, Lecture Notes in Computer Science, Vol. 104 (Springer, Berlin, 1981) 167–183.

[10] L. Pomello, Some equivalence notions for concurrent systems, an overview, in: G. Rozenberg, ed., *Advances in Petri Nets 1985*, Lecture Notes in Computer Science, Vol. 222 (Springer, Berlin, 1986) 381–400.

[11] L. Pomello, Observing net behaviour, in: K. Voss et al., eds., *Concurrency and Nets* (Springer, Berlin, 1987) 403–421.

[12] G. Rozenberg and R. Verraedt, Subset languages of Petri nets, *Theoret. Comput. Sci.* **26** (1983) 301–323.

[13] B.A. Trakhtenbrot and A. Rabinovitch, Behavior structures and nets, *Fund. Inform.* **XI** (1988) 357–404.

[14] W. Vogler, Failures semantics based on interval semiwords is a congruence for refinement, Technical Report TUM-I8905, Technische Universität München, 1989. Extended abstract in: *Proc. STACS'90*, Lecture Notes in Computer Science, Vol. 415, (Springer, Berlin, 1990) 285–297.

[15] W. Vogler, Bisimulation and action refinement, Technical Report SFB-Bericht 342/10/90A, Technische Universität München, May 1990.