Acta Informatica 28, 231–264 (1991)

Concurrent bisimulations in Petri nets*

Eike Best[1], **Raymond Devillers**[2], **Astrid Kiehn**[3], **and Lucia Pomello**[4]

[1] Gesellschaft für Mathematik und Datenverarbeitung, Institut für Methodische Grundlagen, Schloß Birlinghoven, W-5205 St. Augustin 1, Federal Republic of Germany
now at Universität Hildesheim, Institut für Informatik, Samelsonplatz 1, W-3200 Hildesheim, Federal Republic of Germany
[2] Université Libre de Bruxelles, Laboratoire d'Informatique Théorique, Boulevard du Triomphe, B-1050 Bruxelles, Belgium
[3] Technische Universität München, Institut für Informatik, Arcisstrasse 21, W-8000 München, Federal Republic of Germany
now at University of Sussex, Computer Science Department, Falmer, Brighton BN19QH, UK
[4] Università degli Studi di Milano, Dipartimento di Scienze dell'Informazione, Via Moretto da Brescia, 9, I-20133 Milano, Italy

Received July 21, 1989/October 8, 1990

Summary. After various attempts, an equivalence relation is defined for labelled Petri nets, on the base of the concurrency semantics of net theory. This relation, called Fully Concurrent bisimulation and abbreviated FC-bisimulation, preserves the level of concurrency of visible operations and, under some conditions, allows to enforce injective labelling on them. Refinements of a visible operation are also defined and we show that, under some conditions, they preserve FC-bisimulation.

1. Introduction

Bisimulation, which is also sometimes called bisimilarity, has been introduced in [27] as a concept which is essentially equivalent to observational equivalence [25]. Its great importance and usefulness for the comparison of different concurrent systems and for proofs of their correctness has been stressed amply 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. In this paper we apply existing tools of Petri net theory – viz. the description of concurrent behaviour by means of partial orders – in order to propose a new and strengthened definition of bisimulation. The use of Petri nets as a model for concurrent

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

systems allows to display concurrency in an explicit way and supports intuition by means of a simple graphical (thus visual) device; as a counterpart, the formalism offered by Petri nets is not well suited for an algebraic treatment and this may lead, as we shall see, to technically complex developments.

Formally, our definition differs from other proposed approaches [5, 11, 20] in that it is stronger and, under some reasonable restrictions, it is preserved by the refinement of atomic actions. In [17, 34], several concepts have been defined independently, amongst which the BS-bisimulation and the history preserving bisimulation are very close to the notion we propose.

As CCS and CSP can be translated into infinite labelled ordinary Petri nets[1] [2, 10–13] or into finite labelled nets with some additional structures or restrictions [9, 22, 23], it is likely that our definition of FC-bisimulation can be translated easily into those formalisms.

There have been various motivations for this work. One has been the paper [28] in which both the sequence (arbitrary interleaving) and the step sequence (subset sequence) form of various definitions of equivalence were given and compared, resulting in a lattice of implications. While working on augmenting this lattice by giving the various partial order forms of the equivalence definitions (see also [30]), we came up with the one described in this paper. Another origin of this paper was the question whether or not every labelled Petri net can be transformed into an equivalent one for which the labelling is injective on the set of observable actions. [38] contains an extensive discussion why such kinds of labellings are important. Because we thought to have a construction which achieves injectivity, we were looking for as strong a notion of equivalence as possible in order to prove the corresponding result. Finally, another motivation is given by the example discussed in [7]: we were looking for a concept of bisimulation that is preserved by the refinement of atomic actions as discussed in that note.

In Sect. 2 we recollect some basic definitions on plain Petri nets and on transition-labelled nets. In Sect. 3 we re-state bisimulation equivalence and give a different but equivalent definition that is more amenable to proof purposes. Section 4 presents concurrent bisimulation, a first attempt to strengthen usual bisimulation with the use of partial orders, and exhibits the kind of difficulties one has to face in this case. Section 5 contains the definition of fully concurrent (FC-)bisimulation and the proofs that *(i)* it is stronger than ordinary bisimulation, *(ii)* in the sequential case, it is equivalent to ordinary bisimulation, and *(iii)* checking it may be considerably simplified. In Sect. 6, we show that FC-bisimulation preserves the level of concurrency of visible actions, and in particular the absence of self-concurrency. In Sect. 7 we describe a construction that transform every labelled Petri net into an injectively labelled one, and we show that under some conditions the latter is FC-bisimilar to the former. In Sect. 8 we go on to derive conditions for FC-bisimulation to be preserved by the refinement of atomic actions. The refinement notion used in this section has been chosen in order to get interesting results for a large class of Petri nets, allowing non-safeness and the presence of side conditions. Section 9, finally, contains a few concluding remarks.

---

[1] A translation of CCS or CSP into finite ordinary labelled Petri nets is impossible because the former are Turing powerful. On the other hand, the definitions we propose do not severely depend on finiteness

2. Basic definitions

2.1. *Unlabelled systems*

We briefly recall the definitions of some basic concepts, referring e.g. to [4].

- A net with arc weights is a triple $N=(S, T, W)$ where S and T are disjoint sets ($S\cap T=\phi$) and $W: ((S\times T)\cup(T\times S))\to \mathbf{N}=\{0, 1, 2, \ldots\}$.

The transitions in T are generally represented graphically by square boxes, the places in S by circles and the flow function W by weighted arrows.

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 $(x, y)\in((S\times T)\cup(T\times S))$: $W(x, y)\leqq 1$. 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 $x^\bullet=\{y\in S\cup T\mid W(x, y)\neq 0\}$.

The net presents a side condition if there are $s\in S$, $t\in T$ such that $W(s, t)\cdot W(t, s)\neq 0$, i.e. if there is $x\in S\cup T$ such that ${}^\bullet x\cap x^\bullet\neq\phi$.

- A transition t only needs one token *iff* $|{}^\bullet t|=1$ and $W({}^\bullet t, t)=1$.

A state machine net is a net (S, T, W) in which each transition only needs one token and only produces one token, i.e. $\forall t\in T: |{}^\bullet t|=|t^\bullet|=1=W({}^\bullet t, t)=W(t, t^\bullet)$.

- A marking of a net (S, T, W) is defined as a function $M: S\to\mathbf{N}$. A marking is generally represented graphically by depositing tokens in the places.

The transition rule states that a transition t is enabled by M *iff* $M(s)\geqq W(s, t)$ 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* $M(s)\geqq W(s, t_1)+W(s, t_2)$ for all $s\in S$.

More generally, a set $\theta\subseteq T$ is concurrently enabled by a marking M *iff* $M(s)\geqq\sum_{t\in\theta} W(s, t)$ for all $s\in S$. The concurrent occurrence of θ is a step, denoted by $M[\theta\rangle M'$, producing a successor marking M' by the rule $M'(s)=M(s)+\sum_{t\in\theta}[W(t, s)-W(s, t)]$ for all $s\in S$.

Still more generally, a bag $\beta: T\to\mathbf{N}$ is concurrently enabled by a marking M *iff* $M(s)\geqq\sum_{t\in T}\beta(t)\,W(s, t)$ for all $s\in S$ and the concurrent occurrence of β is a general step, denoted by $M[\beta\rangle M'$, producing a successor marking M' by the rule $M'(s)=M(s)+\sum_{t\in T}\beta(t)[W(t, s)-W(s, t)]$ for all $s\in S$.

- A system net (or marked 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 $1\leqq i$.

A sequence $t_1 t_2 \ldots$ is a transition sequence (starting with M) *iff* there is an occurrence sequence $M_0 t_1 M_1 \ldots$ with $M=M_0$. If the finite sequence $t_1 t_2 \ldots t_n$ leads from M to M' then we write $M[t_1 t_2 \ldots t_n\rangle M'$, or simply $M[t_1 t_2 \ldots t_n\rangle$ if we do not want to specify the resulting marking. The set of reachable markings of a marked

net (S, T, W, M_0) is defined as $[M_0\rangle = \{M \mid \exists t_1 t_2 \ldots t_n: M_0[t_1 t_2 \ldots t_n\rangle M\}$. (General) step sequences are defined similarly.

- A marked net (S, T, W, M_0) is 1-safe *iff* $\forall M \in [M_0\rangle \; \forall s \in S: M(s) \leqq 1$.
- 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 $\forall b \in B: |{}^\bullet b| \leqq 1 \wedge |b^\bullet| \leqq 1$ (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. 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), and $\downarrow c$ 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.
- $\mathrm{Min}(N)$ is defined as the set $\{x \in B \cup E \mid {}^\bullet x = \phi\}$, and $\mathrm{Max}(N)$ is defined as the set $\{x \in B \cup E \mid x^\bullet = \phi\}$.
- In order to avoid minor but annoying technical difficulties, we will suppose from now on that all our nets are T-restricted, i.e.: $\forall t \in T: {}^\bullet t \neq \phi \neq t^\bullet$.
- 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 the following three properties (such that π can be interpreted as a concurrent run of Σ): $p(B) \subseteq S$, $p(E) \subseteq T$ (B-elements are instances of place holdings, E-elements are occurrences of transitions); $\mathrm{Min}(N)$ is a B-cut which corresponds to the initial marking M_0, that is, $\forall s \in S: M_0(s) = |p^{-1}(s) \cap \mathrm{Min}(N)|$; and

$$\forall e \in E \; \forall s \in S: \; W(s, p(e)) = |p^{-1}(s) \cap {}^\bullet e| \quad \text{and} \quad W(p(e), s) = |p^{-1}(s) \cap e^\bullet|$$

(transition environments are respected)[2].

The initial process of Σ is the one for which $E = \phi$.

- 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' is an extension of π if π is a prefix of π', i.e. if there is a B-cut c' of π' such that $\Downarrow(\pi', c') = \pi$. The process $\Uparrow(\pi, c)$ is defined analogously. As it turns out, $\Uparrow(\pi, c)$ is a process of $\Sigma' = (S, T, W, M)$ where $M \in [M_0\rangle$ and $\forall s \in S: M(s) = |p^{-1}(s) \cap c|$.

- If a marking M of Σ and a B-cut c of a process π of Σ are such that $\forall s \in S: M(s) = |p^{-1}(s) \cap c|$ then M is said to correspond to c.
- 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 non-empty *Lin*-set. The *Lin*-set of the initial process of a system is simply the empty occurrence sequence.
- 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; i.e., Π is the inverse of *Lin*. It is known that if Σ is 1-safe then $|\Pi(\sigma)| = 1$ for all σ.
- 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, $|Lin(\pi)| = 1$.

[2] For infinite processes there needs to be an additional requirement, but we will not be interested in infinite processes here

2.2. *Labelled systems*

- An alphabet $\mathscr{A}$ is a finite set; we assume that $\tau \notin \mathscr{A}$ (we use τ to denote the 'silent' action).
- A labelling of a net $N=(S, T, W)$ is a function $\lambda: T \to \mathscr{A} \cup \{\tau\}$. λ can be extended in the usual way to a homomorphism $\lambda: T^* \to \mathscr{A}^*$, by combining the classical homomorphic extension of λ with the homomorphism erasing silent labels. The empty word of $\mathscr{A}^*$ is denoted by ε.

In order to avoid ambiguity, it is necessary however to specify if a single transition is considered as belonging to T or to T^*: the result may be τ in the first case and ε in the second one; this will generally be implied by the context.

If $\lambda(t) \in \mathscr{A}$ then t is called observable or external; otherwise, t is called silent or internal.

- $\Sigma=(S, T, W, M_0, \lambda)$ is a labelled system *iff* (S, T, W, M_0) is a marked net and λ is a labelling of (S, T, W).
- We will write $M(w\rangle M'$, where M and M' are markings of a net $N=(S, T, W)$ with a labelling $\lambda: T \to \mathscr{A} \cup \{\tau\}$ and $w \in \mathscr{A}^*$, *iff* there is a transition sequence $t_1 t_2 \ldots t_n$ such that $M[t_1 \ldots t_n\rangle M'$ and $\lambda(t_1 t_2 \ldots t_n)=w$.
- An action $a \in \mathscr{A}$ 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 weaker kind of 1-safeness.)
- 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.

Absence of auto-concurrency is the same as has been termed the 'disjoint' labelling condition' in [33, 29]. It implies the absence of self-concurrency, but not conversely.

3. Bisimulation based on arbitrary interleaving

Let $\Sigma_1=(S^1, T^1, W^1, M_0^1, \lambda^1)$ and $\Sigma_2=(S^2, T^2, W^2, M_0^2, \lambda^2)$ be two fixed labelled net systems. The usual, sequential, definition of bisimulation may be formulated as follows:

Definition 3.1. *Bisimulation based on arbitrary interleaving.*

$\Sigma_1 \approx_{Bis} \Sigma_2$ (Σ_1 is bisimilar to Σ_2) *iff* there is a relation $\rho \subseteq [M_0^1\rangle \times [M_0^2\rangle$ with the following properties:

(i) $(M_0^1, M_0^2) \in \rho$.

(ii) if $(M, M') \in \rho$ then

(a) whenever $M(w\rangle \hat{M}$ for $w \in \mathscr{A}^*$, then $\exists \hat{M}' \in [M_0^2\rangle$: $M'(w\rangle \hat{M}' \wedge (\hat{M}, \hat{M}') \in \rho$;

(b) whenever $M'(w\rangle \hat{M}'$ for $w \in \mathscr{A}^*$, then $\exists \hat{M} \in [M_0^1\rangle$: $M(w\rangle \hat{M} \wedge (\hat{M}, \hat{M}') \in \rho$.

■ 3.1

This means that there are corresponding states (i.e. markings), amongst which the initial ones, such that the visible evolutions from them are the same and lead to corresponding states.

Bisimulation may also be characterized in terms of simple transitions, as in:

Proposition 3.2. *Bisimulation in terms of markings and single transitions.*
$\Sigma_1 \approx_{Bis} \Sigma_2$ *iff there is a relation* $\rho \subseteq [M_0^1\rangle \times [M_0^2\rangle$ *with the following properties:*
(i) $(M_0^1, M_0^2) \in \rho$.
(ii) *if* $(M, M') \in \rho$ *then*
(a) $\forall t \in T^1$, *if* $M[t\rangle \hat{M}$ *then there is an occurrence sequence* $\sigma' = M' t_1' M_1' t_2' M_2' \dots t_n' \hat{M}'$ *such that* $(\hat{M}, \hat{M}') \in \rho$ *and* $\lambda^2(t_1' t_2' \dots t_n') = \lambda^1(t)$.
(b) *Vice versa.*

Proof. We simply have to prove that Requirement (a) in 3.1 *iff* Requirement (a) in 3.2, the case (b) being symmetrical and the rest of the definition being identical.
Now, (a) in 3.1 $\Rightarrow$
$\forall t \in T^1$: $M[t\rangle \hat{M} \Rightarrow M(\lambda^1(t)\rangle \hat{M} \Rightarrow \exists \hat{M}'$ such that $M'(\lambda^1(t)\rangle \hat{M}' \wedge (\hat{M}, \hat{M}') \in \rho$ but $M'(\lambda^1(t)\rangle \hat{M}'$ means that there is a $\sigma' = M' t_1' \dots t_n' \hat{M}'$ with $\lambda^2(t_1' \dots t_n') = \lambda^1(t)$.
Conversely, (a) in 3.2 $\Rightarrow$
whenever $M(w\rangle \hat{M}$, i.e. there is a $\sigma = M t_1 M_1 t_2 M_2 \dots t_n \hat{M}$ with $w = \lambda^1(t_1 \dots t_n)$, then there is a σ_1' such that $M'[\sigma_1'\rangle M_1' \wedge (M_1, M_1') \in \rho \wedge \lambda^2(\sigma_1') = \lambda^1(t_1)$
σ_2' such that $M_1'[\sigma_2'\rangle M_2' \wedge (M_2, M_2') \in \rho \wedge \lambda^2(\sigma_2') = \lambda^1(t_2)$
...
σ_n' such that $M_{n-1}'[\sigma_n'\rangle \hat{M}' \wedge (\hat{M}, \hat{M}') \in \rho \wedge \lambda^2(\sigma_n') = \lambda^1(t_n)$
so that $(\hat{M}, \hat{M}') \in \rho$, $M'(\lambda^2(\sigma_1' \sigma_2' \dots \sigma_n')\rangle \hat{M}'$ and $\lambda^2(\sigma_1' \sigma_2' \dots \sigma_n') = \lambda^2(\sigma_1')\lambda^2(\sigma_2') \dots \lambda^2(\sigma_n')$
$= \lambda^1(t_1)\lambda^1(t_2) \dots \lambda^1(t_n) = \lambda^1(t_1 t_2 \dots t_n) = w$. ■ 3.2

It may be noticed that Definition 3.1 is based on a relation between markings, characterized by the way they are related through occurrence sequences. It is also possible to characterize bisimulation directly in terms of occurrence sequences.

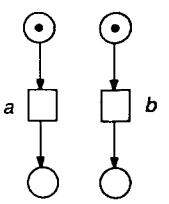
Proposition 3.3. *Bisimulation in terms of occurrence sequences.*
$\Sigma_1 \approx_{Bis} \Sigma_2$ *iff there is a relation* $\bar{\rho} \subseteq \{\sigma \mid M_0^1[\sigma\rangle\} \times \{\sigma' \mid M_0^2[\sigma'\rangle\}$ *with the following properties:*
(i) $(\sigma, \sigma') \in \bar{\rho} \Rightarrow \lambda^1(\sigma) = \lambda^2(\sigma')$ *(same visible traces)*
(ii) $(\varepsilon, \varepsilon) \in \bar{\rho}$ *(empty sequences correspond)*
(iii) *if* $(\sigma, \sigma') \in \bar{\rho}$ *then (extension property)*
(a) *for every occurrence sequence* $\hat{\sigma} = \sigma \sigma_1$, *there is an occurrence sequence* $\hat{\sigma}' = \sigma' \sigma_1'$ *with* $(\hat{\sigma}, \hat{\sigma}') \in \bar{\rho}$.
(b) *Vice versa.*

Proof. $\Rightarrow$
If $\Sigma_1 \approx_{Bis} \Sigma_2$, let us define $\bar{\rho} = \{(\sigma, \sigma') \mid \lambda^1(\sigma) = \lambda^2(\sigma')$ and $(M, M') \in \rho$ if $M_0^1[\sigma\rangle M$ and $M_0^2[\sigma'\rangle M'\}$ where ρ is the relation defined in 3.1;
we clearly have 3.3(i) and 3.3(ii); in order to show 3.3(iii)(a) (and (b), symmetrically), if $(\sigma, \sigma') \in \bar{\rho}$ and $\hat{\sigma} = \sigma \sigma_1$, let $M_0^1[\sigma\rangle M$, $M_0^2[\sigma'\rangle M'$, $M[\sigma_1\rangle \hat{M}$ and $w = \lambda^1(\sigma_1)$; then $M(w\rangle \hat{M}$ and, from the definition of $\bar{\rho}$, $(M, M') \in \rho$; consequently, from 3.1(ii)(a): $\exists \hat{M}'$: $M'(w\rangle \hat{M}' \wedge (\hat{M}, \hat{M}') \in \rho$, but that also means that
$\exists \sigma_1'$: $\lambda^2(\sigma_1') = w = \lambda^1(\sigma_1) \wedge M'[\sigma_1'\rangle \hat{M}'$, and $(\hat{\sigma}, \sigma' \sigma_1') \in \bar{\rho}$.
$\Leftarrow$
If we have a relation $\bar{\rho}$, let us define $\rho = \{(M, M') \mid \exists (\sigma, \sigma') \in \bar{\rho}$: $M_0^1[\sigma\rangle M \wedge M_0^2[\sigma'\rangle M'\}$;
we clearly have 3.1(i), with $\sigma = \varepsilon = \sigma'$;

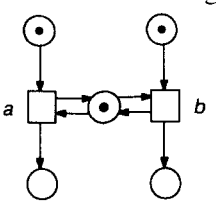
in order to show 3.1(ii)(a) (and (b), symmetrically), if $(M, M') \in \rho$ and $M(w\rangle \hat{M}$, from the definition of ρ there are σ and σ' such that $(\sigma, \sigma') \in \bar{\rho}$, $M_0^1[\sigma\rangle M$ and $M_0^2[\sigma'\rangle M'$, and from the definition of $(\ )$ there is a σ_1 such that $M[\sigma_1\rangle \hat{M}$ and $\lambda(\sigma_1) = w$;
but then, from 3.3(iii)(a) there is a σ_1' such that $(\sigma\sigma_1, \sigma'\sigma_1') \in \bar{\rho}$; from 3.3(i), we have $\lambda^1(\sigma\sigma_1) = \lambda^1(\sigma)\lambda^1(\sigma_1) = \lambda^2(\sigma'\sigma_1') = \lambda^2(\sigma')\lambda^2(\sigma_1')$, so that $\lambda^2(\sigma_1') = \lambda^1(\sigma_1) = w$ since, from $(\sigma, \sigma') \in \bar{\rho}$ and 3.3(i), $\lambda^1(\sigma) = \lambda^2(\sigma')$; and with $M_0^2[\sigma'\sigma_1'\rangle \hat{M}'$, we have $(\hat{M}, \hat{M}') \in \rho$ and $M'(w\rangle \hat{M}'$. ■ 3.3

It may be noticed that, from a practical point of view, characterization 3.3 is more difficult to check than 3.2 since there are many more sequences than markings in general. From a theoretical point of view, on the contrary, the last characterization may be more agreeable, being in a sense more uniform since it is only based on sequences, and not on a mixture of markings and sequences. Moreover, as we shall see, while 3.1, 3.2 and 3.3 are equivalent formulations this will no longer be the case for their concurrent translations. It may also be mentioned that checking 3.3 may be slightly simplified by only considering 1-transition extensions in 3.3(iii), in the same way we went from 3.1 to 3.2. On the other hand, it may also be slightly generalized by allowing infinite sequences.

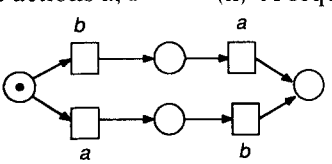
Sequential bisimulation identifies the three systems shown in Fig. 1.



(i) Two concurrent actions a, b



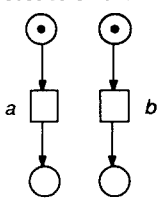
(ii) A sequentialization of (i)



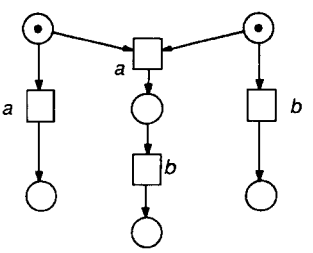
(iii) Another sequentialization of (i)

Fig. 1. Three systems identified by bisimulation

In order to distinguish these systems, it has been proposed ([26, 28, 38]) to generalize the concept of bisimulation by considering (general) step sequences instead of interleavings. However, (general) step sequences fail to discriminate the two systems shown in Fig. 2 for instance.



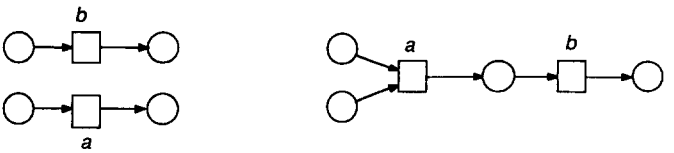
(i) Two concurrent actions ...



(ii) ... and an additional sequential part

Fig. 2. Two systems identified by bisimulation based on step sequences

If we want to be able to state something like 'Σ_1 exhibits exactly as much concurrency as Σ_2' then we should turn to another generalization based on partial orderings, i.e., on the set of processes of a system rather than either occurrence sequences or (general) step sequences. For example, in order to distinguish the two systems Σ_1 and Σ_2 of Fig. 2, we may notice that Σ_2 has a process, namely the one shown in Fig. 3(ii), which is not order-isomorphic (on the set of its observable events) to any process of Σ_1 – since the only process of Σ_1 that contains both a and b is the one shown in Fig. 3(i).



(i) A process of Σ_1 (ii) A process of Σ_2

Fig. 3. Processes of the systems shown in Fig. 2

4. A first attempt to define concurrent bisimulation

In order to formulate bisimulation based on partial orders, we need two auxiliary definitions.

Definition 4.1. *Abstraction of a process of a labelled system.*

Let $\Sigma=(S, T, W, M_0, \lambda)$ with λ: $T\to\mathscr{A}\cup\{\tau\}$ be a labelled system. Let $\pi=(B, E, F, p)$ with p: $B\cup E\to S\cup T$ be a process.

Then the λ-abstraction of π is denoted by $\alpha_\lambda(\pi)=(E', \prec, \lambda')$ and is defined by

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

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

$$\lambda'=\lambda\circ(p|_{E'}), \quad \text{i.e.} \quad \forall e\in E'\colon \lambda'(e)=\lambda(p(e)). \qquad \blacksquare 4.1$$

Of course, $(E', \prec)$ is a partially ordered set, and $\alpha_\lambda(\pi)=(E', \prec, \lambda')$ is thus a labelled poset (with labels in $\mathscr{A}$).

Definition 4.2. *Order-isomorphism of abstractions.*

Let $\alpha_{\lambda_1}=(E'_1, \prec_1, \lambda'_1)$ and $\alpha_{\lambda_2}=(E'_2, \prec_2, \lambda'_2)$ be two abstractions as in the previous definition, both with labels in $\mathscr{A}$. Then $\alpha_{\lambda_1}\cong\alpha_{\lambda_2}$ (they are order-isomorphic) *iff* there is a bijection β: $E'_1\to E'_2$ such that:
(i) $\forall e\in E'_1$: $\lambda'_1(e)=\lambda'_2(\beta(e))$.
(ii) $\forall e_1, e_2\in E'_1$: $e_1\prec_1 e_2$ *iff* $\beta(e_1)\prec_2\beta(e_2)$. $\blacksquare 4.2$

A direct translation of 3.1 could then be

Definition 4.3. *Bisimulation based on partial orders.*

Let $\Sigma_1=(S^1, T^1, W^1, M_0^1, \lambda^1)$ and $\Sigma_2=(S^2, T^2, W^2, M_0^2, \lambda^2)$ be two labelled systems.

$\Sigma_1 \approx_{CB} \Sigma_2$ (Σ_1 and Σ_2 are concurrently bisimilar) *iff* there is a relation $\rho \subseteq [M_0^1\rangle \times [M_0^2\rangle$ with the following properties:

(i) $(M_0^1, M_0^2) \in \rho$.

(ii) if $(M, M') \in \rho$ then:

(a) For every process $\pi_1 = (B_1, E_1, F_1, p_1)$ of the system $(S^1, T^1, W^1, M, \lambda^1)$ there is a process π_2 of the system $(S^2, T^2, W^2, M', \lambda^2)$ such that if $\hat{M}$ and $\hat{M}'$ are the markings corresponding to $\mathrm{Max}(\pi_1)$ and $\mathrm{Max}(\pi_2)$, respectively: $(\hat{M}, \hat{M}') \in \rho$ and $\alpha_{\lambda^1}(\pi_1) \cong \alpha_{\lambda^2}(\pi_2)$
(let us notice that $\mathrm{Min}(\pi_1)$ corresponds to M, and $\mathrm{Min}(\pi_2)$ corresponds to M')

(b) Vice versa. ■ 4.3

This means that there are corresponding states (i.e. markings), amongst which the initial ones, such that the visible concurrent evolutions from them are order-isomorphic and lead to corresponding states.

This definition is similar to some ones proposed in [5, 20] and called pomset bisimulation in [20]. It trivially defines an equivalence relation. Unfortunately, while it may lead to an interesting theory, it does not fit our aims since it does not withstand the simple refinement of an operation into a begin-end sequence, as shown by the example of Fig. 4.

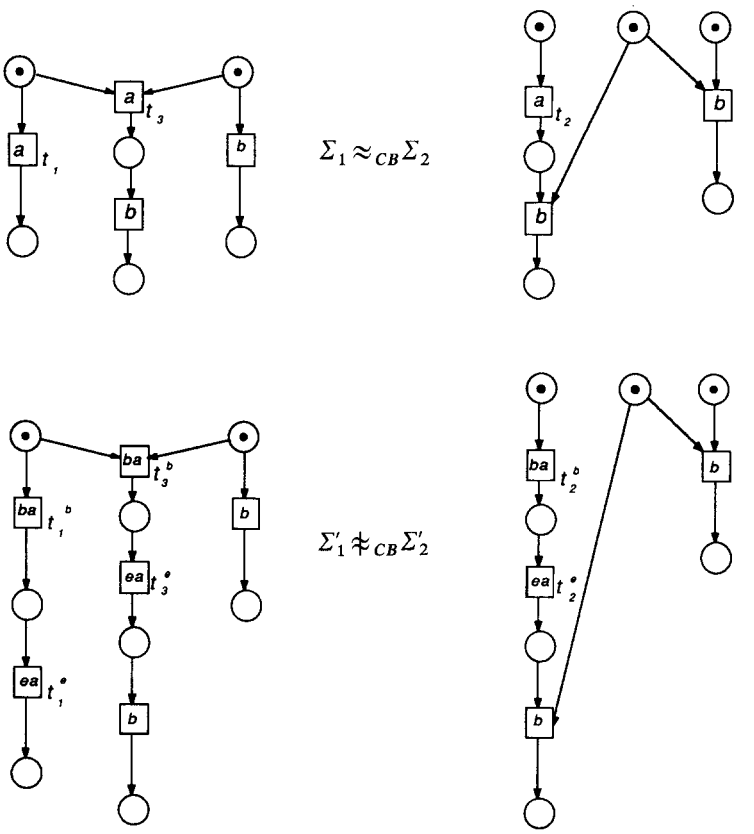
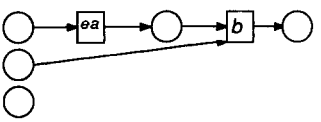
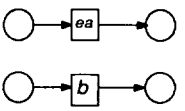


Fig. 4. Two systems where the CB-bisimilarity disappears if the a-labelled operations are replaced by a sequence labelled $b\,a - e\,a$

It may be checked that $\Sigma_1 \approx_{CB} \Sigma_2$ but that the equivalence does not withstand a simple splitting of the a-labelled transitions into two successive ones ($b\,a$ and $e\,a$), as in Σ'_1 and Σ'_2. Indeed, the marking M'_1 resulting from the occurrence of t_1^b in Σ'_1 may only correspond to the marking M'_2 resulting from the occurrence of t_2^b in Σ'_2; but from M'_2 there is a process:

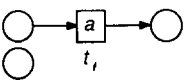


which is not order-isomorphic to the only process with two (visible) transitions from M'_1:

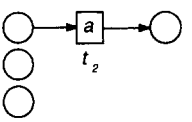


so that $\Sigma'_1 \not\approx_{CB} \Sigma'_2$ (but $\Sigma'_1 \approx_{Bis} \Sigma'_2$).

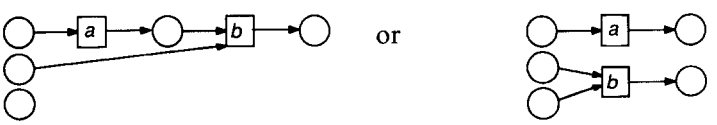
A closer examination of the example shows that the origin of the problem arises from the way partial orders (and processes) concatenate. Indeed, in Σ_1 the process:



yields the marking M_1 which corresponds to the marking M_2 produced in Σ_2 by the process π_2:



From M_2, two processes may occur; their λ-abstractions are both $\boxed{b}$, which is also the λ-abstraction of the only process evolving from M_1, but they differ in the way they concatenate with π_2:



This is of no importance in the checking of the concurrent bisimilarity of Σ_1 and Σ_2 but attains a crucial rôle if the a-labelled transitions are split, as shown for Σ'_1 and Σ'_2.

The problem does not arise with Definition 3.1 since there is only one way to concatenate sequences. The fact that there are in general various ways to concatenate processes may also be related to the reason why, while characterizations 3.2 and 3.3 are equivalent to 3.1, this is no longer true for their translations in terms of processes.

For instance, while concurrent bisimulation is a true refinement of the sequential one, the translation of 3.2 in terms of process evolutions does not add anything to it, as shown by the following two propositions.

Proposition 4.4. *Concurrent bisimulation is a true refinement of sequential bisimulation.*
(a) $\Sigma_1 \approx_{CB} \Sigma_2 \Rightarrow \Sigma_1 \approx_{Bis} \Sigma_2$
(b) $\Sigma_1 \approx_{CB} \Sigma_2 \not\Leftarrow \Sigma_1 \approx_{Bis} \Sigma_2$.

Proof. (a) let us use characterization 3.2; we simply have to prove 3.2(ii)(a);

$$
\begin{aligned}
M[t\rangle \hat{M} &\Rightarrow M[\pi\rangle \hat{M} \text{ where } \pi \text{ only has a } t\text{-transition} \\
&\Rightarrow \exists \pi' \colon M'[\pi'\rangle \hat{M}' \wedge (M, M') \in \rho \wedge \alpha_{\lambda^1}(\pi) \cong \alpha_{\lambda^2}(\pi') \text{ since } \Sigma_1 \approx_{CB} \Sigma_2 \\
&\qquad \text{and } \alpha_{\lambda^1}(\pi) \text{ is the empty order if } t \text{ is a } \tau\text{-transition} \\
&\qquad \text{and a singleton order if } t \text{ is not a } \tau\text{-transition} \\
&\Rightarrow \exists \sigma' \colon M'[\sigma'\rangle \hat{M}' \wedge (\hat{M}, \hat{M}') \in \rho \text{ since a process always has} \\
&\qquad \text{linearizations which are occurrence sequences} \\
&\qquad \text{and } \lambda^2(\sigma') = \varepsilon = \lambda^1(t) \text{ if } t \text{ is a } \tau\text{-transition} \\
&\qquad\qquad = \lambda^1(t) \text{ otherwise}
\end{aligned}
$$

(b) systems Σ'_1 and Σ'_2 in Fig. 4 exhibit an example with $\Sigma'_1 \not\approx_{CB} \Sigma'_2$ but $\Sigma'_1 \approx_{Bis} \Sigma'_2$. ■ 4.4

Proposition 4.5. *Process translation of* 3.2 *is sequential bisimulation.*
$\Sigma_1 \approx_{Bis} \Sigma_2$ *iff there is a relation* $\rho \subseteq [M_0^1\rangle \times [M_0^2\rangle$ *with the following properties:*
(i) $(M_0^1, M_0^2) \in \rho$
(ii) *if* $(M, M') \in \rho$ *then*
  (a) $\forall t \in T^1$, *if* $M[t\rangle \hat{M}$ *then there is a process* π' *from* M' *such that* $M'[\pi'\rangle \hat{M}'$, $(\hat{M}, \hat{M}') \in \rho$ *and* $\alpha_{\lambda^2}(\pi') \cong \lambda^1(t)$
  (b) *Vice versa.*

Proof. Trivial from the observation, already used for 4.4, that if there is at most one non-τ transition, all (partial or total) orders have the same λ-abstractions.

The reason why the proof of 3.2 does not work here results from the fact that in general there are various non-(λ-)isomorphic ways to concatenate (λ-)isomorphic partial orders, while this is not true for total orders, i.e. sequences. ■ 4.5

5. Fully concurrent bisimulation

The analysis of the systems in Fig. 4 suggests that a better, and stronger, definition of bisimulation should include the way the processes concatenate, i.e. how they look from the beginning and how they may be extended. This suggests to start from characterization 3.3 and to translate it in terms of processes and abstraction isomorphisms.

This has been proposed in [14] for instance (but also in [11] in a different context), where it is required that there is a relation between processes such that the initial processes are related, related processes have order-isomorphic

abstractions and any extension of a process π is related to an extension of a process related to π.

We shall here slightly strengthen the formulation by specifying that the extension property concerns not only the processes but also the isomorphisms between their abstractions. This will avoid some awkward behaviours already noticed in [14]. The resulting notion corresponds to the BS-bisimulation introduced in [34] for behaviour structures and to the history preserving bisimulation defined in [17] for event structures.

Definition 5.1. *Fully concurrent bisimulation.*

Let $\Sigma_1=(S^1, T^1, W^1, M_0^1, \lambda^1)$ and $\Sigma_2=(S^2, T^2, W^2, M_0^2, \lambda^2)$ be two labelled systems, $\Sigma_1 \approx_{FCB} \Sigma_2$ *iff* there is a set $\mathscr{B} \subseteq \{(\pi_1, \pi_2, \beta)\}$ such that:

(i) π_1 is a process of Σ_1, π_2 is a process of Σ_2 and β is a relation between the non-τ events of π_1 and π_2.

(ii) if π_0^1 and π_0^2 are the initial processes of Σ_1 and Σ_2, respectively, then $(\pi_0^1, \pi_0^2, \phi) \in \mathscr{B}$.

(iii) if $(\pi_1, \pi_2, \beta) \in \mathscr{B}$, then β is an order-isomorphism between the λ^1-abstraction of π_1 and the λ^2-abstraction of π_2.

(iv) $\forall (\pi_1, \pi_2, \beta) \in \mathscr{B}$:

(a) if π_1' is an extension of π_1, then $\exists (\pi_1', \pi_2', \beta') \in \mathscr{B}$ where π_2' is an extension of π_2 and $\beta \subseteq \beta'$.

(b) Vice versa. ■ 5.1

This clearly defines an equivalence relation. It is also a strengthening of the concurrent bisimulation.

Proposition 5.2. *Fully concurrent bisimulation implies concurrent bisimulation. Let Σ_1 and Σ_2 be two labelled systems: $\Sigma_1 \approx_{FCB} \Sigma_2 \Rightarrow \Sigma_1 \approx_{CB} \Sigma_2$.*

Proof. Let us define $\rho = \{(M_1, M_2) | \exists (\pi_1, \pi_2, \beta) \in \mathscr{B}: M_0^1[\pi_1\rangle M_1 \wedge M_0^2[\pi_2\rangle M_2\}$; 4.3(i) immediately results from 5.1(ii);
Let us now suppose that $(M, M') \in \rho$, i.e. $\exists (\pi, \pi', \beta) \in \mathscr{B}: M_0^1[\pi\rangle M \wedge M_0^2[\pi'\rangle M'$, and $M[\pi_1\rangle \hat{M}$;
as $\mathrm{Max}(\pi)$ and $\mathrm{Min}(\pi_1)$ both correspond to marking M, there is at least one way to concatenate π and π_1, giving a process $\hat{\pi}$ extension of π; consequently, from 5.1(iv)(a), $\exists (\hat{\pi}, \hat{\pi}', \hat{\beta}) \in \mathscr{B}$, $\exists B$-cut c of $\hat{\pi}'$: $\beta \subseteq \hat{\beta}$ and $\Downarrow (\hat{\pi}', c) = \pi'$, so that c corresponds to marking M'; let $\hat{M}'$ be the marking corresponding to $\mathrm{Max}(\hat{\pi}')$: $M_0^2[\hat{\pi}'\rangle \hat{M}'$ and $(\hat{M}, \hat{M}') \in \rho$;
let $\pi_1' = \Uparrow (\hat{\pi}', c)$: $M'[\pi_1'\rangle \hat{M}'$;
from 5.1(iii) and the previous properties, $\hat{\beta} \backslash \beta$ is an order isomorphism between the λ^1-abstraction of π_1 and the λ^2-abstraction of π_1'; hence 4.3(ii)(a), and (b) symmetrically. ■ 5.2

The fact that the strengthening is strict results from the example exhibited in Fig. 4: $\Sigma_1 \approx_{CB} \Sigma_2$ but $\Sigma_1 \not\approx_{FCB} \Sigma_2$.

The fact that it is a strict strengthening of the notion proposed in [14] results from the translation in terms of system nets of Example 5.3 in [17]; as it is slightly lengthy, we shall not include it here. For sequential systems however, as one can imagine, all those notions are equivalent to the sequential bisimulation. This results from the following lemma.

Lemma 5.3. *Processes of sequential systems define total orders.*
If π is a process of a sequential system Σ, the labelled partial order defined on its events is a total order isomorphic to the one defined by its unique corresponding occurrence sequence.

Proof. If two events were not ordered, we know from [3] that there is an evolution of Σ leading to a marking for which the two transitions (not necessarily distinct) corresponding to those events are concurrently enabled, but that would contradict the fact that Σ is sequential. Consequently, the order on the events is total and its labelling corresponds to the unique corresponding transition sequence. ■ 5.3

Proposition 5.4. *The bisimulation definitions collapse for sequential systems.*
If Σ_1 and Σ_2 are two labelled sequential systems,

$$\Sigma_1 \approx_{FCB} \Sigma_2 \Leftrightarrow \Sigma_1 \approx_{CB} \Sigma_2 \Leftrightarrow \Sigma_1 \approx_{Bis} \Sigma_2 .$$

Proof. $\Rightarrow$
results from 5.2 and 4.4
$\Leftarrow$
if $\Sigma_1 \approx_{Bis} \Sigma_2$, i.e. if there is a relation $\bar{\rho}$ satisfying 3.3, then
$\mathscr{B} = \{(\pi_1, \pi_2, \beta) \mid \exists (\sigma_1, \sigma_2) \in \bar{\rho}$: σ_1 corresponds to π_1 and σ_2 to π_2, $\alpha_{\lambda^1}(\pi_1) \cong \alpha_{\lambda^2}(\pi_2)$ and β is such an isomorphism$\}$ satisfies the conditions of 5.1;
indeed, 5.1(ii) trivially results from 3.3(ii),
5.1(iii) results from the definition of $\mathscr{B}$,
we still have to prove 5.1(iv)(a), the case (b) being symmetrical;
from 5.3, $\alpha_{\lambda^1}(\pi_1)$ is total and there is a unique correspondence β_1 between the non-τ events in π_1 and the positions in $w = \lambda^1(\sigma_1)$, and similarly for π_2;
consequently $\beta = \beta_2^{-1} \circ \beta_1$ and is unique;
now, if π_1' is an extension of π_1, its occurrence sequence σ_1' is an extension of σ_1 and from 3.3(iii) there is a $\bar{\rho}$-corresponding extension σ_2' of σ_2; but then, from 3.3(i), $\lambda^1(\sigma_1') = \lambda^2(\sigma_2')$, from [3] there is an extension π_2' of π_2 corresponding to σ_2', and from the previous remark $\alpha_{\lambda^1}(\pi_1') \cong \alpha_{\lambda^2}(\pi_2')$ and the unique isomorphism β' between them is an extension of β. ■ 5.4

Now, in the general case, it is possible to slightly simplify the checking of the extension property by only looking at one-event-extensions.

Proposition 5.5. *Fully concurrent bisimulation from one-event-extensions.*
If Σ_1 and Σ_2 are two labelled systems, $\Sigma_1 \approx_{FCB} \Sigma_2$ iff there is a set $\mathscr{B} \subseteq \{(\pi_1, \pi_2, \beta)\}$ with the following properties:

(i) *π_1 is a process of Σ_1, π_2 is a process of Σ_2 and β is a relation between the non-τ events of π_1 and π_2*
(ii) $(\pi_0^1, \pi_0^2, \phi) \in \mathscr{B}$
(iii) *if $(\pi_1, \pi_2, \beta) \in \mathscr{B}$, then β is an order-isomorphism between $\alpha_{\lambda^1}(\pi_1)$ and $\alpha_{\lambda^2}(\pi_2)$*
(iv) $\forall (\pi_1, \pi_2, \beta) \in \mathscr{B}$
  (a) *if π_1' is an extension of π_1 with only one event more, there is a $(\pi_1', \pi_2', \beta') \in \mathscr{B}$ where π_2' is an extension of π_2 and $\beta \subseteq \beta'$.*
  (b) *Vice versa.*

Proof. Immediate if we observe that any extension of π_1 may be obtained by an iteration of one-event extensions; the corresponding extensions in Σ_2 iteratively obtained will finally yield an extension of π_2 and of β satisfying 5.1(iv)(a); and symmetrically for 5.1(iv)(b). ■ 5.5

In order to convincingly justify Definition 5.1, we still have to exhibit nice properties of the so defined bisimulation notion. That will be the subject of the next chapters.

6. Property preservation

A first characteristic we could ask from a good bisimulation notion is that it should preserve important system properties. Since FC-bisimulation implies the classical sequential one, all the (sequential) properties preserved by latter are trivially also preserved by the former: liveness of any visible operation, generated language, etc. The newly preserved properties should obviously refer to concurrency.

Proposition 6.1. *FC-bisimulation preserves the absence of auto-concurrency. If Σ_1 and Σ_2 are two FC-bisimilar labelled systems and Σ_1 is free of auto-concurrency, then so is Σ_2.*

Proof. Let $\Sigma_2=(S_2, T_2, W_2, M_0^2, \lambda_2)$ and suppose $\exists t, t' \in T_2\ \exists M \in [M_0^2\rangle$: $\lambda(t)=\lambda(t')=a\neq\tau$ and M concurrently enables t, t' in Σ_2.

Then, by known theory [3], there is a finite process π of Σ_2 leading from (a B-cut corresponding to) M_0^2 to (a B-cut corresponding to) M; consequently $\mathrm{Max}(\pi)$ contains two disjoint sets of conditions, one labelled by ${}^\bullet t$ and another one labelled by ${}^\bullet t'$, and there is an extension π_2 of π with two unordered events labelled by t and t', respectively. Clearly, $\alpha_{\lambda^2}(\pi_2)$ has two maximal elements, both labelled by a.

By $\Sigma_1 \approx_{FCB} \Sigma_2$, there is a process $\pi_1=(B, E, F, p)$ of Σ_1 that corresponds to π_2, i.e. $\alpha_{\lambda^1}(\pi_1) \cong \alpha_{\lambda^2}(\pi_2)$.

Hence, $\alpha_{\lambda^1}(\pi_1)$ also contains two maximal elements, both labelled by a. This implies that π_1 contains two concurrent (unordered) events e_1 and e_2, both labelled by transitions of Σ_1 labelled by a. By known theory, one may then find a B-cut of π_1 just before these two events, and the marking corresponding to it enables the two transitions $p(e_1)$ and $p(e_2)$ of Σ_1 (labelled by a) concurrently. This contradicts the hypothesis and finishes the proof. ■ 6.1

Figure 5 shows that $\approx_{Bis}$ is not sufficient to guarantee the last result, since the two systems Σ_1 and Σ_2 of Fig. 5 are $\approx_{Bis}$-equivalent, but Σ_2 contains auto-concurrently while Σ_1 does not.

Σ_1 Σ_2

Fig. 5. Two bisimilar but not FC-bisimilar systems, where only one of them exhibits auto-concurrency

It may also be observed from the proof that what we really proved is slightly more detailed than what we claimed in 6.1.

Proposition 6.2. *FC-bisimulation preserves the auto-concurrent actions.*
If Σ_1 and Σ_2 are FC-bisimilar and an action a is auto-concurrent in Σ_1, the same action is auto-concurrent in Σ_2. ■ 6.2

7. Property enforcement

For some system properties, it is quite natural that they are not preserved by a behavioural equivalence (1-safeness, pureness, no τ-labels, ...). One may then wonder if, on the contrary, the property may be enforced by the equivalence, i.e. if for any system which does not present the property there is an equivalent system which has it, i.e. if in any equivalence class there is a system with the property. This may be true in general or under some reasonable restrictions; it is then a further research subject to look how those restrictions may be dropped. We will here show such a characteristics for the strict labelling property.

Definition 7.1. *Strictly labelled systems.*

A labelled system $\Sigma=(S, T, W, M_0, \lambda)$ with λ: $T\to\mathscr{A}\cup\{\tau\}$ is strictly labelled *iff*

$$\forall t_1, t_2\in T:\ (t_1\neq t_2\wedge\lambda(t_1)\neq\tau\neq\lambda(t_2))\Rightarrow\lambda(t_1)\neq\lambda(t_2)$$

(i.e., λ is injective on non-τ labels). ■ 7.1

In order to get what we want, we shall make use of a call-like construction similar to a construction already used in [23] and, for COSY systems, in [24].

Definition 7.2. *Merging of equally labelled transitions.*

Let $\Sigma=(S, T, W, M_0, \lambda)$ be a labelled system and $A=\{t_1, \ldots, t_k\}\subseteq T$ be a set of transitions with the same label $a\neq\tau$, of cardinality at least 2. We will define a new system $merge(\Sigma, A)=(S', T', W', M'_0, \lambda')$ in the following way:

$S'=S\cup\{in\_A, out\_A\}\cup\bigcup_{t\in A}\{mem\_t\}$

$T'=(T\setminus A)\cup\{A\_called\}\cup\bigcup_{t\in A}\{call\_t, uncall\_t, return\_t\}$

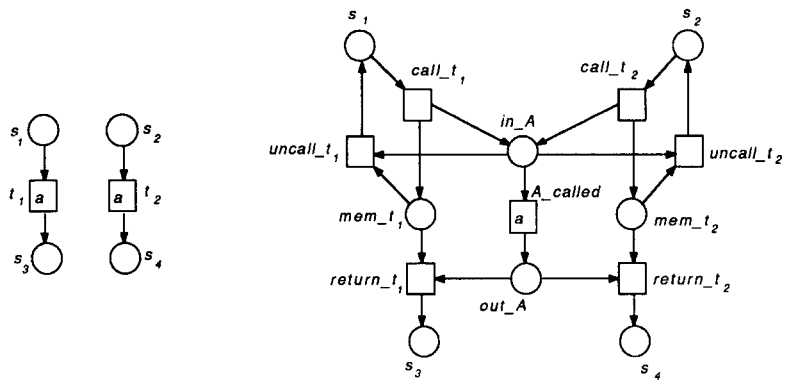
$$W'(x,y)=\begin{cases}W(x,y) & \text{if } x, y\in S\cup(T\setminus A)\\ W(x,t) & \text{if } y=call\_t, \forall t\in A\\ W(y,t) & \text{if } x=uncall\_t, \forall t\in A\\ W(t,y) & \text{if } x=return\_t, \forall t\in A\\ 1 & \text{if } x=call\_t \text{ and } y=in\_A \text{ or } y=mem\_t, \forall t\in A\\ 1 & \text{if } y=uncall\_t \text{ and } x=in\_A \text{ or } x=mem\_t, \forall t\in A\\ 1 & \text{if } y=return\_t \text{ and } x=out\_A \text{ or } x=mem\_t, \forall t\in A\\ 1 & \text{if } x=in\_A \text{ and } y=A\_called,\\ & \text{or } x=A\_called \text{ and } y=out\_A\\ 0 & \text{otherwise}\end{cases}$$

$$M'(x)=\begin{cases}M(x) & \text{if } x\in S\\ 0 & \text{otherwise}\end{cases}$$

$$\lambda'(x)=\begin{cases}\lambda(x) & \text{if } x\in T\cap T'\\ a & \text{if } x=A\_called\\ \tau & \text{otherwise}\end{cases}$$

■ 7.2

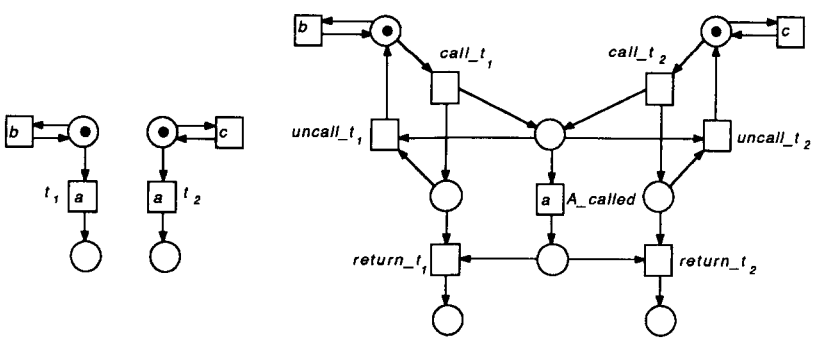
The idea is to introduce for the whole subset A a single *A_called* transition labelled by a, with a single input place *in_A* and a single output place *out_A*; *in_A* has k silent (i.e. τ-labelled) incoming transitions calling for the performance of a and *out_A* has k silent outgoing transitions to continue the work. Additionally, for each of these k silent incoming transitions, a busy wait loop has to be introduced (consisting of another silent *uncall* transition) in order to guarantee bisimulation, as it will be seen later. The definition is sketched for $k=2$ in Fig. 6.



in Σ in $merge(\Sigma, \{t_1, t_2\})$

Fig. 6. Illustration of the Definition 7.2

This definition is a good candidate for transforming any net into a strictly labelled one since, each time it is used, the number of equally labelled visible transitions strictly decreases, and it may be used *iff* the original net is not strictly labelled. Unfortunately, the definition does not always lead to a bisimilar transformed net. A first counterexample is shown in Fig. 7, where the two systems even fail to be language equivalent (hence, they may not be sequentially bisimilar, nor FC-bisimilar of course).



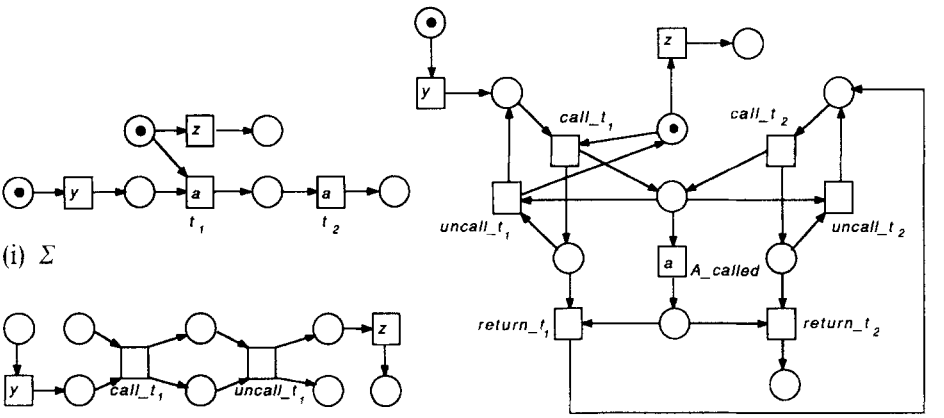
(i) Σ (ii) $\Sigma' = merge(\Sigma, \{t_1, t_2\})$

$\sigma' = call\_t_1\ a\ c\ call\_t_2\ uncall\_t_1\ b$

(iii) A transition sequence of Σ' that has no corresponding one in Σ

Fig. 7. A system whose merging is not bisimilar

The culprit in this case is the fact that the original net presents auto-concurrency, but Fig. 8 exhibits a net without auto-concurrency whose merged version is not FC-bisimilar.



(i) Σ

(ii) $\Sigma' = merge(\Sigma, \{t_1, t_2\})$

(iii) A process of Σ' whose λ'-abstraction has no corresponding one in Σ

Fig. 8. An auto-concurrency free system whose merging is not FC-bisimilar

Here, the problem arises from the fact that, via the busy waiting loop, causal dependencies (between y and z in the example) may be introduced which are not possible in the original system. A more careful examination of the problem shows that this may only happen if one of the merged transitions needs more than one token. It may be observed that in the merged net, the *A_called* transition only needs one token, so that 1-token need is preserved by Construction 7.2.

By combining these remarks one then gets.

Proposition 7.3. *Merging without auto-concurrency and multiple need implies bisimulation.*

If Σ and A are as in 7.2, if no two transitions (not necessarily distinct) of A may be concurrently enabled and if each transition in A only needs one token, then $\Sigma \approx_{FCB} merge(\Sigma, A)$.

Proof. We shall construct a set $\mathscr{B} \subseteq \{(\pi, \pi', \beta)\}$ and show that it fulfils the conditions of characterization 5.5.

(a) Let π' be a process of $\Sigma' = merge(\Sigma, A)$; let $\pi = \pi'' = \pi'$, β = identity relation on the visible events of π' and β'' = identity relation on the conditions of $\mathrm{Max}(\pi')$. We shall progressively modify π and π'', and β and β'' accordingly, in such a way that, at each step:

(a1) β is a bijection between the visible events of π and π', defining an order isomorphism between their $\lambda^{(\prime)}$-abstractions

(a2) π'' is a process of Σ', extension of π' with only silent events in addition, so that π' and π'' have the same $\lambda'^{(\prime)}$-abstractions

(a3) β'' is a bijection between the conditions in $\mathrm{Max}(\pi)$ and those in $\mathrm{Max}(\pi'')$, such that

(a3.1) $\forall c \in \mathrm{Max}(\pi)$: $p(c) = p''(\beta''(c))$
(corresponding maximal conditions have the same label)

(a3.2) $\forall c \in \mathrm{Max}(\pi)$, $\forall e$ visible event in π: $e \prec c \Leftrightarrow \beta''(e) \prec'' \beta(c)$

(in some sense, $\beta'' \cup \beta$ defines an order-isomorphism between the partial orders induced by π and π'' on their visible events and their maximal conditions).

(a4) any event in π has a label in $T \cup T'$; if its label is in $T' \setminus T$, either it has a label *call_t* for some $t \in A$, or it is immediately preceded by a *call_t* labelled event

(a5) the restriction $\tilde{\pi}$ of π to the nodes without predecessors with labels in $T' \setminus T$ is a process of Σ

and finally, π will have no nodes with labels in $T' \setminus T$, so that from (a5) it will be a process of Σ.

Let us first notice that all the properties (a1) to (a5) are initially true; for (a1), (a2) and (a3), it is trivial; for (a4), this results from the observation that, due to the merge definition, in Σ' any transition of the form *uncall_t*, *return_t* (for $t \in A$) and *A_called* needs a token from a place *mem_t* or *in_A*, which is initially empty and may only be filled by a *call_t* transition; for (a5), this results from the fact that the defined restriction only contains events with labels in T with their local environment compatible with Σ, and all the conditions in $\mathrm{Min}(\pi) = \mathrm{Min}(\pi')$ which correspond by definition to the initial marking of Σ and Σ'.

A consequence of (a4) and (a5) will be that, at each step of the construction, if e is an event of π such that: $p(e) \in T' \setminus T$ and $\forall e'$ event of π: $e' \prec e \Rightarrow p(e') \in T$

(i.e. e is minimal among the events of π with labels in $T' \setminus T$),

then $p(e) = call\_t$ for some $t \in A$ and $\forall e''$ event of π: $p(e'') \in T' \setminus T \Rightarrow e \prec e''$ or $e = e''$

(i.e. e is unique and *call_t* labelled).

Indeed, from (a4) and the definition of e', it results that $p(e) = call\_t$ for some $t \in A$, and from the definition of $\tilde{\pi}$ in (a5), it is clear that ${}^\bullet e \subseteq \mathrm{Max}(\tilde{\pi})$.

Now, if e is not unique, that means that there is also e' in π such that $p(e') = call\_t'$, ${}^\bullet e' \subseteq \mathrm{Max}(\tilde{\pi})$, $e' \neq e$, $t' \in A$ (but we may have $t' = t$).

Since, by construction, *call_t* and *call_t'* have the same inputs as t and t', respectively, this means that $\tilde{\pi}$ may be extended in such a way that the result is still a process of Σ, with two maximal events labelled by t and t'; but this means that they are unordered and, from known theory ([3]), that t and t' may be concurrently enabled, which contradicts the hypothesis on A.

Now, the construction proceeds as follows: at each step,

- if π has no labels in $T' \setminus T$, the construction stops
- otherwise, let e be the unique *call_t* labelled event of π which is minimal among the ones with labels in $T' \setminus T$, and let us look at the possible configurations after it. They are sketched in Fig. 9, with the corresponding modifications of π; dotted lines mean there may be a predecessor or a successor; t and t' belong to A and may be identical.

Let us examine each case in turn:

(1) π is modified by dropping the whole configuration but the s-labelled condition c; since this does not affect the visible events and the conditions of $\mathrm{Max}(\pi)$, β, β'' and π'' are not changed; it may easily be verified that all properties (a1) to (a5) are preserved; for instance, if x is a visible event and y is another visible event or a terminal condition: $x \prec y$ before $\Leftrightarrow x \prec y$ after the transformation.

(2) The whole structure is replaced by a single t-labelled event; π'' and β'' are not modified but the event e' will be replaced by e'' in β; it may again be

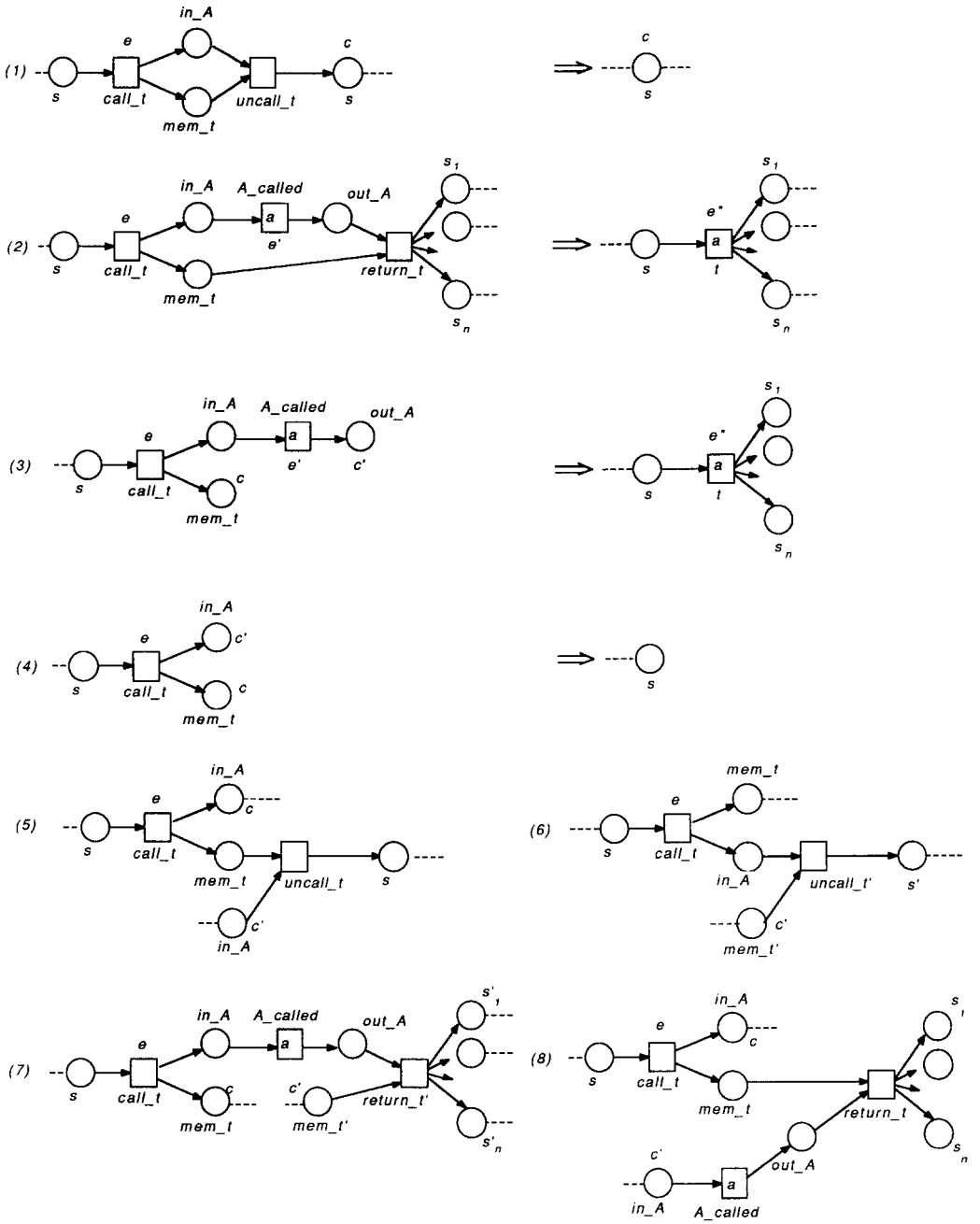


Fig. 9. Possible configurations after the first *call_t* in π

verified that properties (a1) to (a5) are preserved: the partial order on the visible events and on the terminal conditions is not modified, all the events directly following the *call_t* labelled one disappear at once and the new event locally conforms to t in Σ.

(3) The whole structure is again replaced by a single t-labelled event, but the circumstances are quite different here; since e' is the only event following e, there are no other events with labels in $T' \setminus T$ in π and the construction will necessarily stop after this step; if we replace e' by e'' in β, the labelled partial order on the visible events is still the same but as $\mathrm{Max}(\pi)$ is modified we no longer may leave π'' and β'' unchanged. In π'', let us connect the terminal conditions corresponding to c and c' (by β'') to a new event corresponding to *return_t* and let us modify β'' so as to associate corresponding output conditions of it and of e'': it is now easy to check that all properties (a1) to (a5) are again satisfied; moreover, if we now apply the whole construction to π'' instead of π', it is easy to see that we will get exactly the same π, π'', β and β''.

(4) The whole structure collapses in a single condition; here again the construction will stop after this step, β may be left unchanged but we need to modify π'' and β'' since $\mathrm{Max}(\pi)$ has changed. In π'', let us connect the terminal conditions corresponding (by β'') to c and c' to a new event corresponding to *uncall_t* and let us modify β'' so as to associate the new terminal condition of π'' to the new terminal condition of π: it is now easy to check that all properties (a1) to (a5) are again preserved and that if we now apply the whole construction to π'' instead of π, we will get the same π, π'', β, β''.

We may also notice that π'' may only be modified once during the construction: either by a terminal step (3) or (4).

(5) This configuration is not possible. To see this, let us notice that the graph has no cycle and that e precedes any event with a label in $T' \setminus T$, and consequently any condition with a label in $S' \setminus S$. Thus, there must be a path from c to c'; after c, we may have an *uncall* or an *A_called* and a *return* (see Fig. 10, where a *dotted arrow* represents some oriented path).

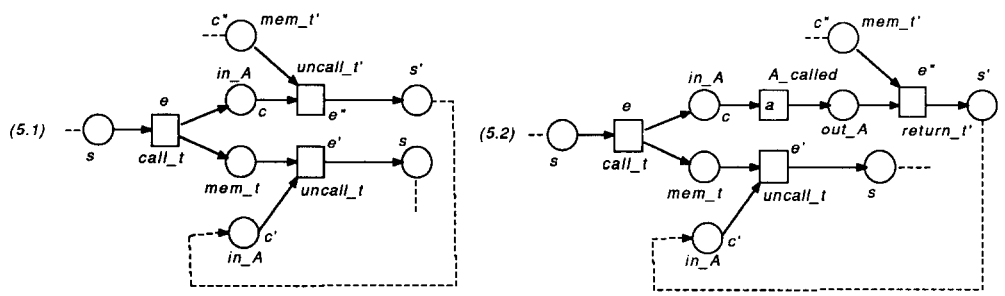


Fig. 10. Further analysis of case (5)

In either case, we must have a path from e to a condition c'', either through e' or e'', but this will always introduce a cycle.

(6) This is again impossible, for similar reasons; the possible configurations are sketched in Fig. 11 and in each case a cycle occurs.

(7) This is again impossible, for similar reasons; the possible configurations are sketched in Fig. 12 and in each case a cycle occurs.

(8) This is again impossible, for similar reasons; the possible configurations are sketched in Fig. 13 and in each case a cycle occurs.

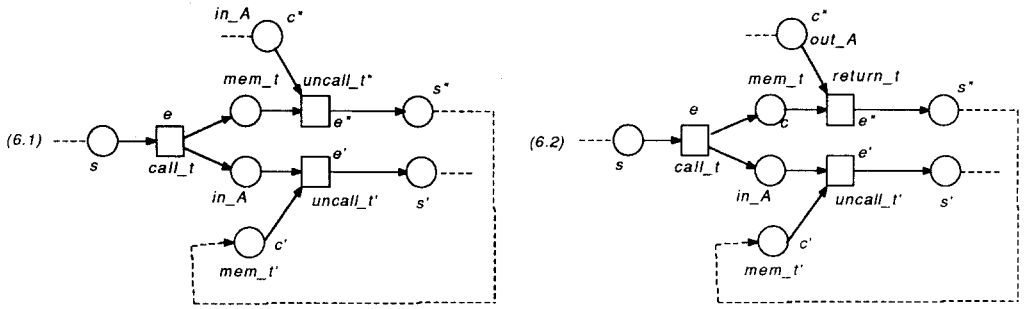


Fig. 11. Further analysis of case (6)

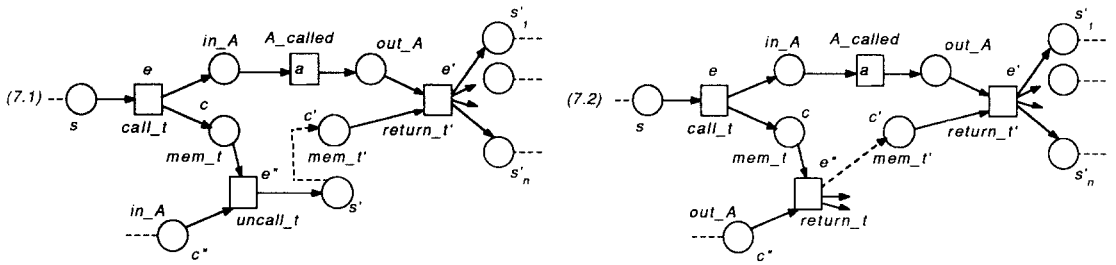


Fig. 12. Further analysis of case (7)

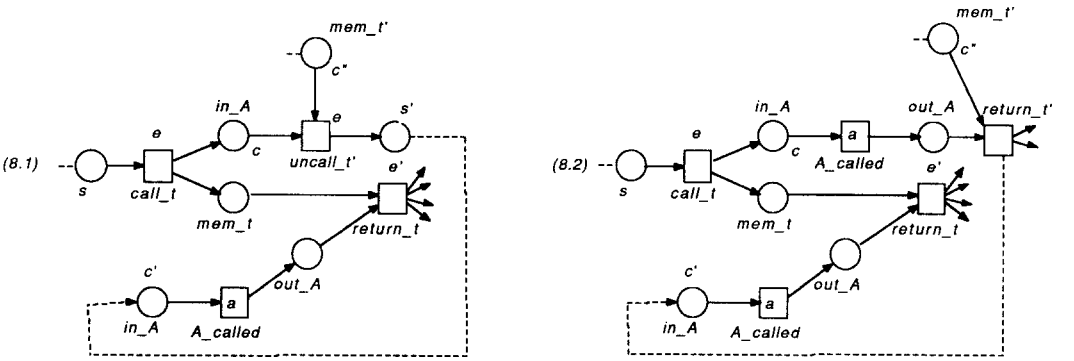


Fig. 13. Further analysis of case (8)

Since, at each step, the number of events with labels in $T' \setminus T$ strictly decreases, the construction will eventually stop.

(b) Let us define $\mathscr{B} = \{(\pi, \pi', \beta) | \pi'$ is a process of Σ', π and β are as constructed in (a)$\}$ and let us check that Σ and Σ' are FC-bisimilar through characterization 5.5:

(i) by definition, π' is a process of Σ' and from the previous construction π is a process of Σ and β is a relation between the visible events of π and π'

(ii) if π' is the initial process of Σ', it is clear that $\pi = \pi'$ is the initial process of Σ, since they have the same initial marking

(iii) from property (a1), the constructed β is an isomorphism between $\alpha_\lambda(\pi)$ and $\alpha_{\lambda'}(\pi')$

(iv) ● let $\tilde{\pi}$ be an extension of π with only one additional event e; let π'' be the extension of π' constructed with π and β from π', and let β'' be

the order preserving bijection between $\mathrm{Max}(\pi)$ and $\mathrm{Max}(\pi'')$ also defined during this construction.
If the label of e is not in A, if we add an identical event c'' to π'', connected to the conditions of $\mathrm{Max}(\pi'')$ corresponding through β'' to the input conditions of e, we will get an extension $\tilde{\pi}'$ of π such that, if we apply the construction to it, we will get $\tilde{\pi}$ and $\tilde{\beta}=\beta$ (if e is not visible) or $\tilde{\beta}=\beta\cup\{(e, e'')\}$ (otherwise); this is due to the way the construction works and to the order preserving property of β''.
If the label of e is $t\in A$, we may do the same trick while replacing e'' by a whole (*call_t*, *A_called*, *return_t*) structure.

- Let $\tilde{\pi}'$ be an extension of π' with only one additional event e, and let us apply the construction (a) to $\tilde{\pi}'$; if e has a label *call_t*, *uncall_t* or *return_t* for some $t\in A$, we will get the same π and β as for π' (so that we have $\tilde{\pi}=\pi$ and $\tilde{\beta}=\beta$): the construction is the same up to the end where a step (4) is added, the terminal step (4) is replaced by a terminal step (1) or the terminal step (3) is replaced by a terminal step (2), respectively;
if e has the label *A_called*, the construction will be the same as for π' up to the terminal step (4) which is replaced by the terminal step (3), so that $\tilde{\pi}$ will indeed be an extension of π and $\tilde{\beta}=\beta\cup\{(e', e'')\}$;
if e has a label in $T\cap T'$, it is connected to conditions with labels in S only, which also belongs to π and π'' and for which β'' is still the identity; consequently, if we apply the construction to $\tilde{\pi}'$, it will be the same as for π' and at the end e will be connected at the same conditions, so that $\tilde{\pi}$ is indeed an extension of π, and $\tilde{\beta}=\beta$, if e is visible, or $\tilde{\beta}=\beta\cup\{(e, e)\}$, otherwise.

This terminates the proof. ■ 7.3

As an immediate corollary, one then gets

Proposition 7.4. *Every system without auto-concurrency and multiple inputs is FC-bisimilar to a strict one.*
Let $\Sigma=(S, T, W, M_0, \lambda)$ be a labelled system such that, for any visible label occurring more than once, any transition with this label only needs one token and no two (not necessarily distinct) transitions with this label may be concurrently enabled. Then, there exists a strictly labelled system Σ' for which $\Sigma\approx_{FCB}\Sigma'$.

Proof. For any visible label a occurring more than once in Σ, let $A=\{t\in T\mid\lambda(t)=a\}$; from 7.3 and 6.2, *merge*(Σ, A) is FC-bisimilar to Σ and the conditions of the theorem are preserved, so that we may resume the construction until the labelling is strict. In fact, since *A_called* in the definition of *merge* only needs one token, instead of merging all the transitions with the same label, we may also use any subset with at least two transitions and resume the construction. ■ 7.4

It may be observed that the problem exhibited on Fig. 8 does not arise if we only look at the interleaving semantics, i.e. if we use the usual sequential bisimulation. Consequently, it is not too surprising that in this case we have a simpler result:

Proposition 7.5. *Merging without auto-concurrency implies interleaving bisimulation.*
If Σ and A are as in 7.2 and if no two transitions (not necessarily distinct) of A may be concurrently enabled, then $\Sigma\approx_{Bis}$ merge(Σ, A).

Proof. As this does not concern concurrent bisimulation, we shall only give a sketch of the proof here:

(a) $\forall M' \in [M'_0\rangle$: $M'(in\_A) + M'(out\_A) = \sum_{t \in A} M'(mem\_t)$

indeed, this is true initially and any transition occurrence maintains this relation;

(b) since no two transitions of A may be concurrently enabled, we also have that $\forall M' \in [M'_0\rangle$: $M'(in\_A) + M'(out\_A) \leqq 1$. Consequently, from (a) there is at most one token in at most one of the *mem_t* places in any reachable marking of $merge(\Sigma, A)$;

(c) let us define

$\rho = \{(M, M') \mid M \in [M_0\rangle, M' \in [M'_0\rangle$ and $\forall s \in S$:

$$M(s) = M'(s) + \begin{cases} W(s,t) & \text{if } M'(in\_A) = 1 = M'(mem\_t) \\ W(t,s) & \text{if } M'(out\_A) = 1 = M'(mem\_t) \} \\ 0 & \text{otherwise} \end{cases}$$

it is rather easy to check that ρ satisfies all the requirements of 3.2. ■ 7.5

As an immediate corollary, one then gets

Proposition 7.6. *Every system without auto-concurrency is bisimilar to a strict one.*
Let $\Sigma = (S, T, W, M_0, \lambda)$ be a labelled system such that, for any visible label occurring more than once, no two (not necessarily distinct) transitions with this label may be concurrently enabled. Then, there exists a strictly labelled system Σ' for which $\Sigma \approx_{Bis} \Sigma'$.

Proof. For any visible label a occurring more than once in Σ, let $A = \{t \in T \mid \lambda(t) = a\}$; from 7.5 and the relation between the reachable markings, $merge(\Sigma, A)$ is bisimilar to Σ and the conditions of the theorem are preserved, so that we may resume the construction until the labelling is strict. ■ 7.6

8. Transition refinement

Another natural question about an equivalence relation concerns its stability over some operation or transformation rule, i.e. its congruence (or not). In our context, this 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 the 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 multiple of the original ones in order to homogenize the characteristics. In a first draft of the present paper, we have used such

a definition, but we soon found that we were confronted with severe problems in the general case, which includes non-safeness and side conditions; some of these problems will be discussed later. In [18], a corresponding definition can be found which works for 1-safe side condition free nets.

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 8.1. *Empty in/out system.*
A labelled system $D=(S^D, T^D, W^D, M_0^D, \lambda^D)$ is 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) $M_0^D(s_{in})=1$ and $\forall s\neq s_{in}: M_0^D(s)=0$
$\forall M\in[M_0^D\rangle: M(s_{out})>0\Rightarrow[M(s_{out})=1\wedge\forall s\neq s_{out}: M(s)=0]$
(i.e. initially there is a unique token in s_{in} and at the end there is a unique token in s_{out})
(iii) $\forall t\in s_{in}^{\bullet}: W^D(s_{in}, t)=1$ and $\forall t\in{}^{\bullet}s_{out}: W^D(t, s_{out})=1$
(s_{in}, resp. s_{out}, represents the whole set of tokens consumed, resp. produced, by any refined transition). ■ 8.1

Definition 8.2. *Empty in/out refinement.*
Let $\Sigma=(S, T, W, M_0, \lambda)$ be a labelled system with $\lambda: T\to\mathscr{A}\cup\{\tau\}$.
Let $D=(S^D, T^D, W^D, M_0^D, \lambda^D)$ be an empty in/out system with $\lambda^D: T^D\to\mathscr{A}^D\cup\{\tau\}$.
Let $a\in\mathscr{A}\cap cod(\lambda)$ be a label occurring in Σ.

The refinement $ref(\Sigma, a, D)$ is (up to isomorphism) the labelled system $(S', T', W', M_0', \lambda')$ where
(a) $S'=S\cup\{\langle s, t\rangle \mid s\in S^D\setminus\{s_{in}, s_{out}\}, t\in\lambda^{-1}(a)\}$
(b) $T'=(T\setminus\lambda^{-1}(a))\cup\{\langle x, t\rangle \mid x\in T^D, t\in\lambda^{-1}(a)\}$
(c)

$$W'(x', y')=\begin{cases} W(x', y') & \text{if } x', y'\in S\cup(T\setminus\lambda^{-1}(a)) \\ W^D(x, y) & \text{if } x'=\langle x, t\rangle, y'=\langle y, t\rangle \text{ for some } t\in\lambda^{-1}(a) \\ W(x', t) & \text{if } y'=\langle y, t\rangle, x'\in{}^{\bullet}t \text{ for some } t\in\lambda^{-1}(a) \text{ and } y\in s_{in}^{\bullet} \\ W(t, y') & \text{if } x'=\langle x, t\rangle, y'\in t^{\bullet} \text{ for some } t\in\lambda^{-1}(a) \text{ and } x\in{}^{\bullet}s_{out} \\ 0 & \text{otherwise} \end{cases}$$

(d)

$$M'(s)=\begin{cases} M(s) & \text{if } s\in S \\ 0 & \text{otherwise} \end{cases}$$

(e)

$$\lambda'(t')=\begin{cases} \lambda(t') & \text{if } t'\in T\setminus\lambda^{-1}(a) \\ \lambda^D(x) & \text{if } t'=\langle x, t\rangle \text{ for some } x\in T^D, t\in\lambda^{-1}(a). \end{cases}$$ ■ 8.2

This definition corresponds to the following construction

Construction 8.3. *Construction of an empty in/out refinement.*
Let $\Sigma=(S, T, W, M_0, \lambda)$ be a labelled system with $\lambda: T\to\mathscr{A}\cup\{\tau\}$.
Let $D=(S^D, T^D, W^D, M_0^D, \lambda^D)$ be an empty in/out system with $\lambda^D: T^D\to\mathscr{A}^D\cup\{\tau\}$.
Let $a\in\mathscr{A}\cap cod(\lambda)$ be a label occurring in Σ.

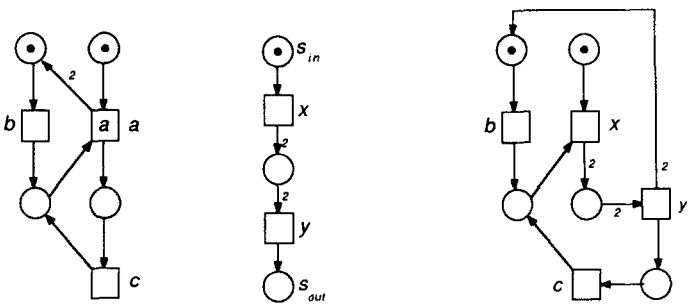
The refinement $ref(\Sigma, a, D)$ is (up to isomorphism) the labelled system obtained from Σ by applying the following construction for each $t \in \lambda^{-1}(a)$ (the order does not matter):

(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.
$\forall x, y \in S^D \cup T^D$: $W(\langle x, t\rangle, \langle y, t\rangle) = W^D(x, y)$, $\forall x \in T^D$: $\lambda(\langle x, t\rangle) = \lambda^D(x)$
and $\forall x \in S^D \setminus \{s_{in}, s_{out}\}$: $M_0(\langle x, t\rangle) = 0$

(iii) connect the successors of s_{in} to the predecessors of t and the predecessors of s_{out} to the successors of t:
$\forall x \in s_{in}^\bullet$, $\forall y \in {}^\bullet t$: $W(y, \langle x, t\rangle) = W(y, t)$
$\forall x \in {}^\bullet s_{out}$, $\forall y \in t^\bullet$: $W(\langle x, t\rangle, y) = W(t, y)$. ■ 8.3

It may be noticed that, at the end, $ref(\Sigma, a, D)$ will be a labelled system with λ: $T \to (\mathscr{A} \setminus \{a\}) \cup \mathscr{A}^D \cup \{\tau\}$. Figure 14 shows an example of this construction.



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

Fig. 14. Illustration of Definition 8.2

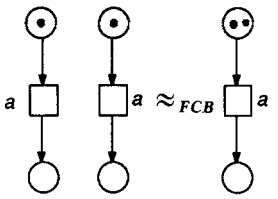
Because of their restrictions, Definitions 8.1 and 8.2 work in the general case, particularly also if side conditions are present and if the net is not 1-safe. However, these restrictions are weak enough to encompass

- classical subcases like simple splitting, simple choice and renaming (which we will address later)
- the case that silent input and output interface transitions are used to connect the refinement system copies; if such silent transitions are incorporated in the refinement system then we conform with the conditions described in 8.1.

Even so, rather awkward situations may occur, as exhibited in Figs. 15 and 16.

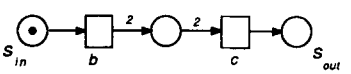
A careful examination of the example in Fig. 15 shows that the problem here arises from the combination of the fact that a refined transition is self-concurrent and that in the refinement system there is a transition needing more than one token. But excluding this is not enough.

In the example of Fig. 16, which is similar to an example devised in [21] for process graphs, the systems are 1-safe, self-concurrency free and each transition needs and produces only one token. A closer examination shows that the culprit in this case is quite surprisingly, the presence of a silent transition after a refined transition. This may also be related, as it is done in [21], to the observation that, in the Hennessy-Milner axiomatization of the (weak) bisimulation equiva-



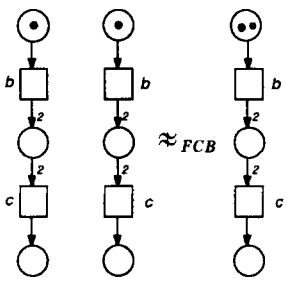
Σ_1 Σ_2

(i) two FC-bisimilar labelled systems



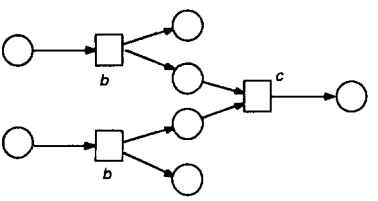
D

(ii) an empty in/out system



$ref(\Sigma_1, a, D)$ $ref(\Sigma_2, a, D)$

(iii) the refinements of Σ_1 and Σ_2



(iv) a process of $ref(\Sigma_2, a, D)$ which has no corresponding process in $ref(\Sigma_1, a, D)$

Fig. 15. A first (counter) example

lence for process algebras, the third τ-law $a\cdot(\tau\cdot x+y)=a\cdot(\tau\cdot x+y)+a\cdot x$ is not preserved by refinements and exactly gives our example with $x=nil$ and $y=b\cdot nil$.

Combining these observations then leads us to the following results.

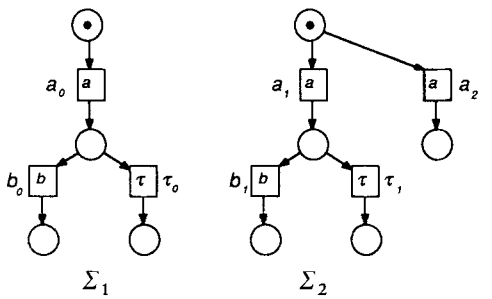
Definition 8.4. *SM-system.*

An SM-system is an empty in/out system D such that (S^D, T^D, W^D) is a state machine net.

■ 8.4

Proposition 8.5. *FC-bisimulation is preserved by SM-refinements in the absence of silent transitions immediately after the refined transitions.*
Let $\Sigma_1=(S^1, T^1, W^1, M_0^1, \lambda^1)$ *and* $\Sigma_2=(S^2, T^2, W^2, M_0^2, \lambda^2)$ *be two labelled systems,*
let $D=(S^D, T^D, W^D, M_0^D, \lambda^D)$ *be an SM-system and*
let $a\in cod(\lambda^1)\cap cod(\lambda^2)$; *if* $\forall t, t'\in T^1: \lambda^1(t)=a\wedge t'\in t^{\bullet\bullet}\Rightarrow\lambda^1(t')\neq\tau$
and $\forall t, t'\in T^2: \lambda^2(t)=a\wedge t'\in t^{\bullet\bullet}\Rightarrow\lambda^2(t')\neq\tau$,
then $\Sigma_1\approx_{FCB}\Sigma_2\Rightarrow\Sigma_1'=ref(\Sigma_1, a, D)\approx_{FCB}\Sigma_2'=ref(\Sigma_2, a, D)$.

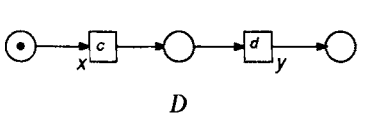
Proof. The basic observation here is the fact that for any SM-system, all the processes are simple chains, i.e. nets such that each node has exactly one predecessor (except for the unique minimal condition) and one successor (except for the unique maximal condition). Consequently, from a set $\mathcal{B}=\{(\pi_1, \pi_2, \beta)\}$ with the properties explained in 5.1 or 5.5, characterizing the fact that Σ_1 and Σ_2



(i) two FC-bisimilar labelled systems

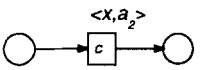
0	0
a_0	a_1
$a_0 b_0$	$a_1 b_1$
$a_0 \tau_0$	$a_1 \tau_1$
$a_0 \tau_0$	a_2

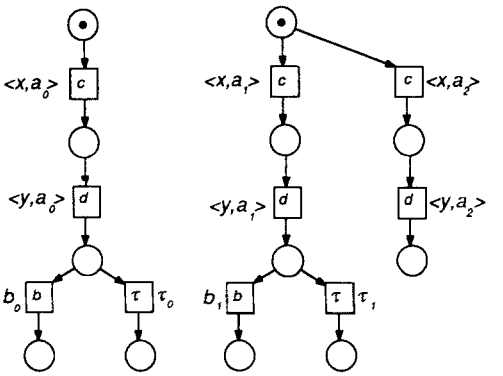
(ii) the corresponding processes



D

(iii) an empty in/out system





$ref(\Sigma_1, a, D)$ $ref(\Sigma_2, a, D)$

(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 $ref(\Sigma_1, a, D)$

Fig. 16. A second (counter) example

are FC-bisimilar, it is rather easy to construct a set $\mathscr{B}' = \{(\pi'_1, \pi'_2, \beta')\}$ with the same properties showing that Σ'_1 and Σ'_2 are also FC-bisimilar.

Let π'_1 be any process of Σ'_1. Any node of π'_1 whose label does not belong to $S^1 \cup T^1$:

- has a label of the form $\langle x, t\rangle$ for some $t \in (\lambda^1)^{-1}(a)$
- 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 $z \in T^D \cup S^D \setminus \{s_{\text{in}}, s_{\text{out}}\}$.

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 node in the chain (which must then be an event with a label $\langle u, t\rangle$ where u is an immediate predecessor of s_{out} in D) (case 1)

unless the chain stops on a maximal condition before (case 2).

Consequently, for each such chain γ, we may replace it
- in the first case, by an event $e(\gamma)$ with the label t since they have the same inputs and outputs
- in the second case, by an event $e(\gamma)$ with the label t and new output conditions corresponding to t, since they have the same inputs and there is nothing afterwards: in this case, $e(\gamma)$ is a maximal event.

The resulting object will be some process π_1 of Σ_1; let π_2 and β be such that $(\pi_1, \pi_2, \beta) \in \mathscr{B}$ and let us apply the following construction:

For any γ constructed previously, let us replace in π_2 the event $\beta(e(\gamma))$, with its label t_2, by a copy of the chain γ where all the labels $\langle x, t_1 \rangle$ are replaced by $\langle x, t_2 \rangle$, and drop the output conditions of $\beta(e(\gamma))$ if the chain is uncomplete, i.e. if it terminates on a condition; this is possible since in that case $e(\gamma)$ was a maximal event and, since there are no silent transitions immediately after $\beta(e(\gamma))$ and β is order preserving, so is $\beta(e(\gamma))$; in the other case, $\beta(e(\gamma))$ and the chain have the same outputs (from t_2) and in both cases they have the same inputs (from t_2 again).

Now, it should be clear that the constructed object, π'_2, is a process of Σ'_2 and that, if we define
$\beta' = \{(e_1, e_2) \in \beta \mid \lambda^1(p^1(e_1)) \neq a \neq \lambda^2(p^2(e_2))\}$
$\cup \{(e_1, e_2) \mid e_1$ has a label $\langle x, t_1 \rangle$, e_2 has a label $\langle x, t_2 \rangle$, x is visible in D, e_1 and e_2 are at the same position in their chains and t_1 and t_2 are related as explained above$\}$,
β' is an order isomorphism between π'_1 and π'_2.

It may also be observed that if we apply the construction from π'_2 in the reverse direction, we will get exactly the same π'_1 and β'.

If π'_1 is the initial process of Σ'_1, we may obtain as π'_2 the initial process of Σ'_2 since then, in the construction, $\pi_1 = \pi_0^1$ and we may choose $\pi_2 = \pi_0^2$.

Let us now consider the extension property 5.5(iv):

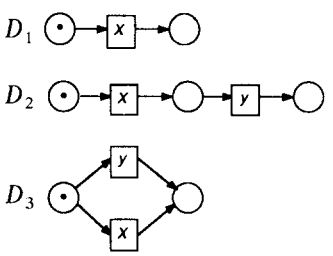
if we add an event e'_1 to π'_1, three cases are possible:
- it may be in a chain γ, i.e. e'_1 has a label $\langle x, t_1 \rangle$, and the construction will lead to π'_2 with exactly one more event, with a label $\langle x, t_2 \rangle$, in the copy of γ
- it may be the beginning of a new chain γ, and the construction will lead to π'_2 extended with possibly some τ events and the start of a new chain corresponding to the other one since, due to the absence of silent transitions immediately after a-labelled transitions, they both correspond to maximal a-labelled events in the non-refined processes
- it may be in no chain; it may only be connected to full chains and non-a events; then, the construction will again lead to an extension of π'_2 where the extension part is exactly the same as in the corresponding non-refined process;

if we add an event e'_2 to π'_2, due to the symmetry of the construction, the situation is the same.
Consequently, all the conditions of the characterization 5.5 are fulfilled. This terminates the proof. ■ 8.5

As an immediate corollary, we get

Corollary 8.6. *Renaming, simple splitting and simple choice replacement.*
If there is no silent transition, FC-bisimulation is preserved by renaming, simple splitting and simple choice replacement.

Proof. This results from the fact that



are SM-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. ■ 8.6

By attacking the other reason of the failure in Fig. 15, we get

Proposition 8.7. *FC-bisimulation is preserved by empty in/out refinements in self-concurrency free environments, in the absence of silent transitions immediately after the refined transitions.*
Let $\Sigma_1=(S^1, T^1, W^1, M_0^1, \lambda^1)$ *and* $\Sigma_2=(S^2, T^2, W^2, M_0^2, \lambda^2)$ *be two labelled systems, let* $D=(S^D, T^D, W^D, M_0^D, \lambda^D)$ *be an empty in/out system and let* $a\in cod(\lambda^1)\cap cod(\lambda^2)$ *be a label such that no a-labelled transition is self-concurrent, if* $\forall t,\ t'\in T^1$: $\lambda^1(t)=a\wedge t'\in t^{\bullet\bullet}\Rightarrow\lambda^1(t')\neq\tau$ *and* $\forall t,\ t'\in T^2$: $\lambda^2(t)=a\wedge t'\in t^{\bullet\bullet}\Rightarrow\lambda^2(t')\neq\tau$
then
$\Sigma_1\approx_{FCB}\Sigma_2\Rightarrow\Sigma_1'=ref(\Sigma_1, a, D)\approx_{FCB}\Sigma_2'=ref(\Sigma_2, a, D)$.

Proof. We shall re-use here some of the techniques we already exploited in the proofs of 7.3 and 8.5. From a set $\mathscr{B}=\{(\pi_1, \pi_2, \beta)\}$ with the properties explained in 5.5, characterizing the fact that Σ_1 and Σ_2 are FC-bisimilar, we shall construct a set $\mathscr{B}'=\{(\pi_1', \pi_2', \beta')\}$ with the same properties showing that Σ_1' and Σ_2' are also FC-bisimilar.

Let π_1' be any process of Σ_1'. Any node of π_1' whose label does not belong to $S^1\cup T^1$ (if any) has a label of the form $\langle x, t\rangle$ for some $t\in(\lambda^1)^{-1}(a)$ and has a predecessor with a label $\langle y, t\rangle$ where y is an immediate successor of s_{in} in D.

Let e_1 be a minimal node whose label does not belong to $S^1\cup T^1$ (if there is no such label, the construction stops); e_1 is an event with a label $\langle y, t_1\rangle$ where ${}^\bullet y=\{s_{in}\}$ and $t_1\in(\lambda^1)^{-1}(a)$; moreover there is no other event with the same property and the same t_1 since the set of nodes without predecessors with a label out of $S^1\cup T^1$ is a process of Σ_1, ${}^\bullet e_1$ is in the Max of this process and if there were two such e_1's corresponding to the same t_1, transition t_1 would be self-concurrent in Σ_1 since they use exactly the same tokens. Consequently, e_1 is a predecessor for all the nodes with a label of the form $\langle x, t_1\rangle$. Let γ be the maximal structure issued from e_1, where all the nodes have a label $\langle z, t_1\rangle$: e_1 is the unique minimal element amongst them.

Now, two cases are possible:

- either γ does not contain any event with a label $\langle u, t_1\rangle$ 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 ($\neq e_1$) in γ has some of its input conditions out of γ; but any of them has a label of the form $\langle v, t_1\rangle$, is a successor of e_1 and there should be a path from e_1 to that condition leaving γ at some point; this could only be through an event with a label $\langle w, t_1\rangle$ where $w \in {}^\bullet s_{out}$, but we supposed there were no such event; 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_1, and the adequate output conditions: $e(\gamma)$ is then a maximal event and the construction may resume, with another (possibly the same) t_1;

- or γ contains one or more events e'_1 with a label $\langle u, t_1\rangle$ where $u \in {}^\bullet s_{out}$; if γ has no event (but e_1) with some input conditions out of γ, then, up to the initial condition and the terminal ones corresponding to the various e'_1's, γ is a process of D; but then, from the definition of an empty in/out refinement, when u is reached there are no token left in D, so that e'_1 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_1 and the unique output event e'_1; we may then replace γ by a new event $e(\gamma)$, with a label t_1, since the inputs and outputs of γ correspond to those of t_1, and resume the construction since we are again in the adequate conditions;

now, if there is an event ($\neq e_1$) in γ with input conditions out of γ, let c be a minimal condition of this type; c has a label $\langle v, t_1\rangle$ and is a successor of e_1; there is thus a path from e_1 to c leaving γ at some point: this may only be at some e'_1; since c is minimal, no event between e_1 and e'_1 may have inputs out of γ but then, between e_1 and e'_1, 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.

At the end, it should be clear that the resulting object is some process π_1 of Σ_1; let π_2 and β be such that $(\pi_1, \pi_2, \beta) \in \mathscr{B}$ and, for any γ constructed previously, let us replace in π_2 the event $\beta(e(\gamma))$, with its label t_2, by a copy of γ where all the labels $\langle x, t_1\rangle$ are replaced by $\langle x, t_2\rangle$, and drop the output conditions of $\beta(e(\gamma))$ if γ is uncomplete (second case); the last part of the construction is possible since in that case $e(\gamma)$ is maximal and, due to the fact that there are no silent transitions immediately after it and β is order preserving, so is $\beta(e(\gamma))$.

It should be clear that the constructed object, π'_2, is a process of Σ'_2 and that, if we define $\beta' = \{(e_1, e_2) \in \beta \mid \lambda^1(p^1(e_1)) \neq a \neq \lambda^2(p^2(e_2))\} \cup \{(e_1, e_2) \mid e_1$ has a label $\langle x, t_1\rangle$, x is visible in D and e_2 is a copy of $e_1\}$, β' is an order isomorphism between π'_1 and π'_2.

It may also be observed that if we apply the construction from π'_2 in the reverse direction, we will get exactly the same π'_1 and β'.

The initial processes of Σ'_1 and Σ'_2 trivially correspond to each other and the extension property 5.5(iv) is fulfilled since

if we add an event e'_1 to π'_1,

- it may be in a structure γ, and the construction will lead to π'_2 with exactly one more event in the copy of γ, at the same position
- it may be the beginning of a new structure γ, and the construction will lead to π'_2 with the start of a new chain corresponding to the other one since, due again to the absence of silent transitions immediately after a-labelled transitions, they both correspond to maximal a-labelled events in the non-refined processes

• it may be in no such structure; it may then be connected to complete structures and non-a events, and the construction will again lead to an extension of π'_2, where the extension part is exactly the same as in the corresponding non-refined process
if we add an event e'_2 to π'_2, due to the symmetry of the construction, the situation is the same.

Consequently all the conditions of characterization 5.5 are fulfilled. ■ 8.7

Concerning the application domain of this property 8.7, let us notice that a 1-safe system net is automatically self-concurrency free.

It may be pointed out that other ways are possible to devise equivalence notions withstanding refinements. For instance, one could introduce in the semantics of actions the fact that they have a begin and an end, like in [1, 35], and this indeed leads to equivalences preserved by split-like refinements. But here, we wanted to find an equivalence based on partial orders which has the branching structure of bisimulations and keeps atomicity of actions. Moreover, as shown by the example in Fig. 15, in our more general context where we want to allow non 1-safe systems as much as possible, it may happen that two systems are split-bisimilar but that their equivalence disappears for more general refinements.

As a last point, let us illustrate the kind of difficulties one could face if one tries to extend the theory even to (apparently) elementary systems which are not empty in/out ones, with a multiplicative interface. This is exhibited in Fig. 17.

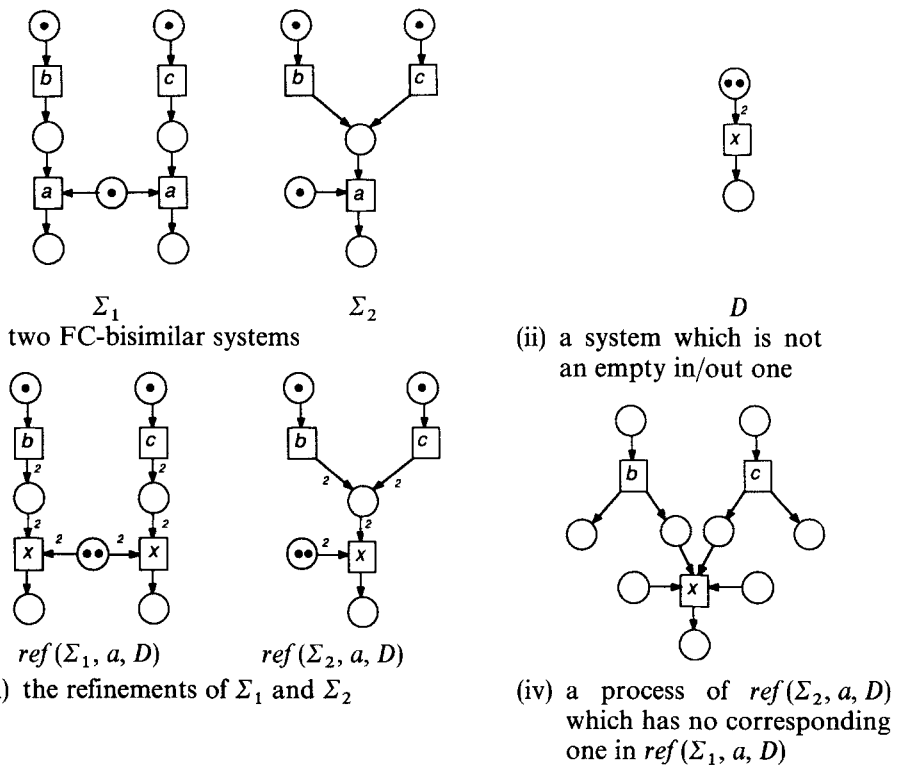


Fig. 17. FC-bisimulation does not withstand elementary non-empty-in/out refinements

9. Conclusion

We have proposed a bisimulation definition based on partial orders for the system model of transition-labelled Petri nets. We have shown that this notion enjoys some desirable characteristics: it preserves some system properties, may enforce some other ones, and is a congruence with respect to refinement.

Our definition essentially coincides with BS-bisimulation ([34]) and history preserving bisimulation ([17, 19]) developed independently by other authors in the context of different system models.

The enforcement result we have proved in this paper is probably unique. The congruence result, on the other hands, has a similar flavour as other results reported recently in the literature. However, the result of [17] concerns a more restricted system model and a different notion of refinement, slightly relaxed in [19]; the result of [16] concerns a weaker notion of bisimulation and a different system model, while [35] deals with a different kind of equivalence. Further results of this type are reported in [1, 6, 8], for instance.

While these results are all more or less different, it has become apparent more recently that there may be some convergence, and not only through system-to-system translations. Within the context of event structures, [36] suggests that there may be a general principle behind forging a given equivalence relation into a congruence which is as weak as possible. Furthermore, in order to deal with the counterexamples we have given, it will be necessary to modify the notion of bisimulation. In fact, [15] and [36] have reported that the definition we give in this paper (and history preserving bisimulation) can be extended to deal satisfactorily with examples such as that of Fig. 16 and, more generally, with the presence of invisible actions. However, we believe that many more pieces of work like these are necessary before a general and uniform theory of equivalences and refinements could be expected to emerge. For (labelled) Petri nets, it is equally useful to consider place (state) refinements, and the question of congruence arises in the respect; however, much less work has been reported in this context; for what concerns state observability one may mention for instance [32], and for what concerns place refinements see [37]; the interested reader may also consult the reviews [6, 31].

Acknowledgments. We are very much indebted to Antoni Mazurkiewicz and Walter Vogler for their careful reading and constructive remarks, and to the anonymous referees who helped us in improving the presentation of this paper.

References

1. Aceto, L., Hennessy, M.: Towards action-refinement in process algebras. Report 3/88, Computer Science Department, University of Sussex (1988). Also in Proc. LICS 89 (Asilomar, California), IEEE Computer Society Press, pp. 138–145 (1989)
2. Best, E.: COSY: Its relation to nets and to CSP. In: Brauer, W., et al. (eds.) Advances in Petri Nets 1986. Proc. of Advanced Course on Petri Nets, Bad Honnef (1986) (Lect. Notes Comput. Sci., vol. 255, pp. 216–220). Berlin Heidelberg New York: Springer 1987
3. Best, E., Devillers, R.: Sequential and concurrent behaviour in Petri net theory. TCS **55**, No. 1 (1988)
4. Best, E., Fernández, C.: Notation and terminology on Petri net theory. Arbeitspapiere der GMD Nr. 195 (1987)

5. Boudol, G., Castellani, I.: On the semantics of concurrency: Partial orders and transition systems. In: Ehrig, H. (ed.) Proceedings TAPSOFT 87, Vol. 1. (Lect. Notes Comput. Sci., vol. 249, pp. 123–137). Berlin Heidelberg New York: Springer 1987
6. Brauer, W., Gold, R., Vogler, W.: Behaviour and equivalence preserving refinements of Petri Nets. Draft paper, Techn. Univ. München, submitted to Advances in Petri Nets 1990 (Lect. Notes Comput. Sci.). Berlin Heidelberg New York: Springer (accepted for publication)
7. Castellano, L., de Michelis, G., Pomello, L.: Concurrency vs. interleaving: an instructive example. Bull. EATCS **31**, 12–15 (1987)
8. Darondeau, Ph., Degano, P.: Event structures, causal trees and refinements. In: Rovan, B. (ed.) Proc. of Mathematical Foundations of Computer Science 90. (Lect. Notes Comput. Sci., vol. 452, pp. 237–245). Berlin Heidelberg New York: Springer 1990
9. de Cindio, F., de Michelis, G., Pomello, L., Simone, C.: A Petri net model of CSP. Proc. CIL'81, Barcelona (1981)
10. de Cindio, F., de Michelis, G., Pomello, L., Simone, C.: Milner's communicating systems and Petri Nets. In: Pagnoni, A., Rozenberg, G. (eds.), Application and Theory of Petri Nets (Inf. Fachber., vol. 66, pp. 40–59). Berlin Heidelberg New York: Springer 1983
11. Degano, P., de Nicola, R., Montanari, U.: Observational equivalences for concurrency models. Formal descr. of programming concepts III. Wirsing, M. (ed.), pp. 105–132. Amsterdam: North Holland 1987
12. Degano, P., de Nicola, R., Montanari, U.: A distributed operational semantics for CCS based on condition/event systems. Acta Inf. **26**, 59–91 (1988)
13. Degano, P., Gorrieri, R., Marchetti, S.: An exercise in concurrency: A CSP process as a condition/event system: In: Rozenberg, G. (ed.) Advances in Petri Nets 1988. (Lect. Notes Comput. Sci., vol. 340, pp. 85–105). Berlin Heidelberg New York: Springer 1988
14. Devillers, R.: On the definition of a bisimulation notion based on partial words. Petri Net Newsletter **29**, 16–19 (1988)
15. Devillers, R.: Maximality preserving bisimulation. Technical Report LIT-214, Univ. Bruxelles (1990)
16. van Glabbeek, R.: The refinement theorem for ST-bisimulation semantics. Proc. IFIP Working Group Conference on Programming Concepts and Methods. Broy, M., Jones, C.B. (eds.) (to appear 1990)
17. van Glabbeek, R., Goltz, U.: Equivalence notions for concurrent systems and refinement of actions. Arbeitspapiere der GMD 366 (1989). Extended abstract in: Kreczmar, A., Mirkowska, G. (eds.) Mathematical Foundations of Computer Science. (Lect. Notes Comput. Sci., vol. 379, pp. 237–248). Berlin Heidelberg New York: Springer 1989
18. van Glabbeek, R., Goltz, U.: Refinement of actions in causality based models. In: de Bakker, J.W., de Roever, W.P., Rozenberg, G. (eds.). Stepwise refinement of distributed systems. Models, formalisms, correctness (Lect. Notes Comput. Sci., vol. 430, pp. 267–300). Berlin Heidelberg New York: Springer 1990
19. van Glabbeek, R., Goltz, U.: Equivalence notions and refinement of actions for flow event structures. In DEMON deliverables, Esprit BRA 3148 (1990)
20. van Glabbeek, R., Vaandrager, F.: Petri net models for algebraic concurrency. Proc. of PARLE Conference. In: de Bakker, J.W., Nijman, A.J., Treleaven, P.C. (eds.) PARLE. Parallel Architectures and Languages Europe. Vol. 2. (Lect. Notes Comput. Sci., vol. 259, pp. 224–242). Berlin Heidelberg New York: Springer 1987
21. van Glabbeek, R., Weijland, W.: Refinement in branching time semantics. Proceedings of the International Conference on Algebraic Methodology and Software Technology – Iowa City, USA (1989)
22. Goltz, U.: On representing CCS programs by finite Petri nets. Arbeitspapiere der GMD (1988)
23. Goltz, U., Reisig, W.: CSP programs as nets with individual tokens (Lect. Notes Comput. Sci., vol. 188, pp. 169–196). Berlin Heidelberg New York: Springer 1985
24. Lauer, P.E., Campbell, R.: Formal semantics of a class of high-level primitives for coordinating concurrent processes. Acta Inf. **5**, 297–332 (1975)
25. Milner, R.: A calculus of communicating systems (Lect. Notes Comput. Sci., vol. 92). Berlin Heidelberg New York: Springer 1980
26. Nielsen, M., Thiagarajan, P.S.: Degrees of non-determinism and concurrency: A Petri net

view. 4th Conf. on Found. of Softw. Techn. and Theor. Comp. Science, pp. 89–117. Berlin Heidelberg New York: Springer 1984
27. Park, D.: Concurrency and automata on finite sequences. Computer Science Department, University of Warwick (1981)
28. Pomello, L.: Some equivalence notions for concurrent systems. An overview. In: Rozenberg, G. (ed.) Advances in Petri nets 1985 (Lect. Notes Comput. Sci., vol. 222, pp. 381–400). Berlin Heidelberg New York: Springer 1986
29. Pomello, L.: Observing net behaviour. In: Voss, K., et al. (eds.) Concurrency and nets, pp. 403–421. Berlin Heidelberg New York: Springer 1987
30. Pomello, L.: Osservatore, reti di Petri, Processi. Ph.D. Thesis, University of Milano and Torino – Italy (1988)
31. Pomello, L., Simone, C.: A survey of equivalence notions for net based systems. Draft paper, Univ. Milano. In: Rozenberg, G. (ed.) Advances in Petri nets 1991 (Lect. Notes Comput. Sci.). Berlin Heidelberg New York: Springer (to be published)
32. Pomello, L., Simone, C.: A state transformation preorder over a class of EN systems. In: Rozenberg, G. (ed.) Advances in Petri nets 1990 (Lect. Notes Comput. Sci.). Berlin Heidelberg New York: Springer 1990
33. Rozenberg, G., Verraedt, R.: Subset languages of Petri nets (Lect. Notes Comput. Sci., vol. 66) (also: TCS 26, pp. 301–323, 1983). Berlin Heidelberg New York: Springer 1978
34. Rabinovitch, A., Trakhtenbrot, B.A.: Behaviour structures and nets. Fundamenta Informaticae XI, 357–404 (1988)
35. Vogler, W.: Failures semantics based on interval semiwords is a congruence for refinement. In: Choffrut, C., Lengauer, T. (eds.) Proc. STACS 90 (Lect. Notes Comput. Sci., vol. 415, pp. 285–297). Berlin Heidelberg New York: Springer 1990
36. Vogler, W.: Bisimulation and action refinement. SFB-Bericht 342/10/90 A, Techn. Univ. München (1990)
37. Vogler, W.: Failures semantics of Petri nets and the refinement of places and transitions. Technical Report TUM-I9003, Techn. Univ. München (1990)
38. Voss, K.: System specification with labelled nets and the notion of interface equivalence. Arbeitspapiere der GMD 211 (1986)