Structural Petri net equivalence

Eindhoven University of Technology
Department of Mathematics and Computing Science

Structural Petri Net Equivalence

by

M. Voorhoeve

96/07

ISSN 0926-4515

editors: prof.dr. R.C. Backhouse
prof.dr. J.C.M. Baeten

Reports are available at:
http://www.win.tue.nl/win/cs

Computing Science Report 96/07
Eindhoven, March 1996

Structural Petri Net Equivalence

M. Voorhoeve (email: wsinmarc@win.tue.nl)

Eindhoven University of Technology

Abstract

An equivalence relation, called structural bisimilarity, is given for labeled Place-Transition nets. In contrast to behavioral bisimilarity of nets, this equivalence only depends upon the structure of the nets considered, that is, their places and transitions and the way these are connected. It does not involve any conversion to transition systems. Algorithms are given for reducing a net to its normal form and deciding whether two given nets are bisimilar. The paper concludes by giving a behavioral characterization of structural net bisimilarity. A given net induces an ordered process space by considering markings as states, steps as transitions and bag inclusion as the partial ordering. Structural bisimilarity of nets is equivalent to order-preserving (noninterleaving) bisimilarity of their induced process spaces.

Keywords: Petri Nets, Concurrency, Bisimulation, Graph algorithms.

1 Introduction

A Petri net is a bipartite directed graph. The nodes of a net are divided into passive nodes, called *places* and active ones, called *transitions* [1]. The edges are divided into *consumption* edges leading from places to transitions and *production* edges leading from transitions to places. Places that have a consumption (production) edge leading to (from) a transition are called *input* (*output*) places of that transition. A Petri net can be *marked* with *tokens* in its places.

Marked Petri nets can model the behavior of concurrent dynamic systems. The state of such a system is represented by the marking of the net. A step of the system consists of the concurrent *firing* of one or more transitions. The step can occur if the transitions in it are *enabled*, i.e. they all can consume enough tokens from their input places. If the step occurs, the enabling tokens are *consumed* from the input places and new tokens are *produced* for the output places. This leads to a new marking (state).

Marked Petri nets are widely used in concurrency theory. They can be represented graphically and their behavior can be easily understood. Constructions exist for operators in languages like ACP and CCS: sequential composition, choice, interleaved merge and concurrent merge (c.f. [GlVa87]). Causal subtleties like confusion, disjunctive causality and resolved conflict (see [Gla95]) that cannot be expressed in ACP and CCS are expressible in the behavior of Petri nets. Since the constructions for sequential composition and choice are somewhat cumbersome, Petri nets are being regarded as a kind of "assembly language" of concurrency, providing models for "higher" alge-

[1] The term "transition" in Petri net theory deviates from common usage. Petri transitions can *bring about* state changes (ordinary transitions). We call state changes "steps" as much as possible to avoid confusion.

braic languages. As a consequence, the states of nets are abstracted from and reduced to mere "causes" for future actions. An exponent of this approach is the Petri Box Calculus [BeDH92].

However, this use of Petri nets does not fully exploit its potential. High-level Petri nets are being used to describe large-scale systems in a concise and accessible way. With tools like Design/CPN ([Jen92]) and ExSpect ([HeSV91]), Petri nets have become a specification language as well! Advantages of such a direct Petri net based modeling are the graphical representation and accessability for non-specialists.

Another advantage is the possibility to describe an externally visible state. In fact, states (bags of places) are the dual of steps (bags of transitions). Many concurrent systems possess a state that can be observed by its environment and is used to base decisions upon. However, what can be observed from the state is in general only a *projection* of the actual system's state, just like what is observed from a step is a projection of the actual step taking place within the system. An ACP-based approach that incorporates states in a similar way can be found in [BaBe95].

Making the distinction between observable and actual states and steps can be done by labeling places and transitions. The labels constitute the observable part of a state or step. Places and transitions with the same label may cause different states or steps to be observed as the same. By relabeling the net, thus identifying some states and steps, one obtains projection or abstraction. Often, one wants to verify that some projection of a complex system (the implementation) is equivalent to some simple system (the specification).

In this respect, Petri net based modeling suffers from a major drawback when compared to algebraic languages. These languages possess operators for renaming and abstraction, with a rich and powerful equivalence theory. Such a theory is less well developed for Petri nets, possibly because of its assembly language status. One way of catching up is to convert nets into algebraic process terms, like in [BaVo95]. However, this approach amounts to computing the occurrence graph, so equivalence relations stemming from it may be undecidable, even for finite nets. Note that finite nets can have infinite behavior. Also the state of a net is disregarded. Other approaches, viz. [PoRS92] have similar drawbacks.

In this paper, the more ambitious approach is taken of developing a purely Petri net oriented equivalence theory, that does not require any translation from the net to a transition system. A bisimulation oriented equivalence relation for Petri nets is defined that is decidable for finite nets. Algorithms are given for normalizing a net and deciding bisimilarity of nets. It is proved that this structural bisimilarity corresponds to order preserving bisimilarity for the (ordered) process spaces induced by the net. The relation thus preserves behavioral properties, like liveness and boundedness. We assert that it also preserves many structural properties.

The advantages of this novel approach for Petri net analysis are manifold. Instead of analyzing a given net, its (smaller) normal form can be taken as a starting point for analysis. Most important, however, is the theoretical foundation of an equivalence theory that takes both actions and states into account.

The paper starts with a notation section. In a section on Petri nets we give our structural equivalence relation and establish some basic properties. A next section is devoted to an algorithm that normalizes a given net. The following sections give the promised behavioral characterization of structural net bisimulation and a small example. The final section discusses the practical meaning

of structural net bisimilarity.
The author wishes to acknowledge Andries Brouwer for sharing his insights in graph algorithms and Twan Basten for many fruitful discussions and suggestions for improvement.

2 Basic notions

Let A, B, C be sets. $A \times B$ is the set of ordered pairs (a, b) with $a \in A$ and $b \in B$. $\mathbb{P}(A \times B)$ is the set of relations between A and B. Let $R \in \mathbb{P}(A \times B)$ be such a relation. For $a \in A$; $b \in B$ we write $a\ R\ b$ iff $(a, b) \in R$. The *domain* $\mathrm{dom}(R)$ of R is defined as the set $\{a \in A \mid \exists b \in B : a\ R\ b\}$. The inverse R^{-1} of R is defined by $b\ R^{-1}\ a \Leftrightarrow a\ R\ b$. R is *total* iff $\mathrm{dom}(R) = A$ and *functional* iff $\forall a \in A; b, b' \in B : a\ R\ b \wedge a\ R\ b' \Rightarrow b = b'$. R is *injective* iff R^{-1} is functional and *surjective* iff R^{-1} is total. Functional relations are called functions. The set of total functions within $\mathbb{P}(A \times B)$ is denoted $A \to B$. For $f \in A \to B$; $a \in A$, the unique element b such that $a\ f\ b$ is named $f(a)$. A bijective relation or bijection is functional, total, injective and surjective. The *composition* $R \circ S$ of relations $R \in \mathbb{P}(A \times B)$ and $S \in \mathbb{P}(B \times C)$ is defined by $\forall a \in A; c \in C : a\ R \circ S\ c \Leftrightarrow \exists b \in B : (a\ R\ b \wedge b\ S\ c)$. The *transitive closure* R^+ of a relation $R \in \mathbb{P}(A \times A)$ is the relation $R \cup (R \circ R) \cup (R \circ (R \circ R)) \cup \ldots$.
The set $\mathbb{B}(A)$ of bags with elements from A is the set of total functions in $A \to \mathbb{N}$. For $\beta \in \mathbb{B}(A)$; $a \in A$, $\beta(a)$ is called the *multiplicity* of a in β. We write $a \in \beta$ iff $\beta(a) > 0$. β is called *finite* iff the set $\{a \in A \mid a \in \beta\}$ is finite. A finite bag β is denoted by juxtaposing the elements $a \in \beta$, superscripted with their multiplicity like a^1 (a singleton bag) or $a^2b^1c^3$. The empty bag is denoted $\mathbf{0}$. The *size* of a finite bag $a_1^{k_1} \ldots a_n^{k_n}$ is $k_1 + \ldots + k_n$. For $\alpha, \beta \in \mathbb{B}(A)$, bag addition and subtraction is defined by $\forall a \in A : (\alpha + \beta)(a) = \alpha(a) + \beta(a)$ and $\forall a \in A : (\alpha - \beta)(a) = \max(0, \alpha(a) - \beta(a))$. For $n > 0$; $\alpha \in \mathbb{B}(A)$, we write $n.\alpha$ for $\alpha + \ldots + \alpha$ iterated n times. By definition, $0.\alpha = \mathbf{0}$. If $I = \{i_1 \ldots, i_n\}$ is a finite set and $\alpha_i \in \mathbb{B}(A)$ for $i \in I$, then we write $\Sigma_{i \in I} \alpha_i$ for $\alpha_{i_1} + \ldots + \alpha_{i_n}$. $\Sigma_{i \in \emptyset} \alpha_i$ equals $\mathbf{0}$ by definition. If $I = i_1^{k_1} \ldots i_n^{k_n}$ is a finite bag and $\alpha_i \in \mathbb{B}(A)$ for $i \in \mathrm{dom}(I)$, then we write $\Sigma_{i \in I} \alpha_i$ for $k_1.\alpha_{i_1} + \ldots + k_n.\alpha_{i_n}$. $\Sigma_{i \in \mathbf{0}} \alpha_i$ equals $\mathbf{0}$ by definition. The partial order $\leq$ on bags is defined by $\alpha \leq \beta \Leftrightarrow \alpha - \beta = \mathbf{0}$. This is called *bag inclusion*. We set $\alpha < \beta \Leftrightarrow \alpha \leq \beta \wedge \alpha \neq \beta$ (strict bag inclusion).
For $R \in \mathbb{P}(A \times B)$, the relation $\hat{R} \in \mathbb{P}(\mathbb{B}(A) \times \mathbb{B}(B))$ relates bags iff the elements of these bags can be related by R. $\hat{R}$ is defined as the minimal (w.r.t. set inclusion) relation satisfying $\mathbf{0}\ \hat{R}\ \mathbf{0}$, $a^1\ \hat{R}\ b^1 \Leftrightarrow a\ R\ b$ and $\alpha_1\ \hat{R}\ \beta_1 \wedge \alpha_2\ \hat{R}\ \beta_2 \Rightarrow \alpha_1 + \alpha_2\ \hat{R}\ \beta_1 + \beta_2$. If R is a function, so is $\hat{R}$.

3 Labeled P/T nets

We presuppose an alphabet $\mathcal{L}$ of *labels*. A labeled place-transition net (or LPT net) is a bipartite directed graph (with multiple edges allowed) having labeled nodes.

Definition 1 *An LPT net N is a four-tuple (S, T, F, ℓ), where S, T are disjoint sets (of places and transitions respectively); $F \in \mathbb{B}((T \times S) \cup (S \times T))$ is the weighted flow relation and $\ell \in (T \cup S) \to \mathcal{L}$ is the labeling function. The pairs (d, e) in $\mathrm{dom}(F)$ with $F(d, e) > 0$ are called*

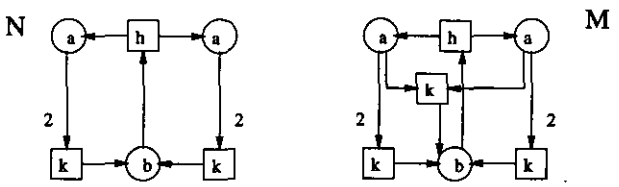

Figure 1: Example nets

arcs *and* $F(d,e)$ *is the* weight *of such an arc. Given a net* N*, its components are denoted by subscripting them with* N*.*

For a given net, the *cause* and *effect* functions $C, E \in T \to \mathbb{B}(S)$ are defined as follows. For $t \in T$, $C(t) = \Sigma_{p \in S} F(p,t).p^1$ and $E(t) = \Sigma_{p \in S} F(t,p).p^1$.

In many cases the weights in the flow relation are 1. For this class of LPT nets, the definitions of F, C and E can be simplified, involving sets instead of bags.

The most distinctive equivalence relation on LPT nets is isomorphism. Two nets are isomorphic iff their nodes can be mapped 1-1 to one another while preserving node labels and edges with their weights. Isomorphism abstracts from the sets S and T of an LPT net by considering the labels and the net structure alone.

Definition 2 *An* isomorphism *between LPT nets* N *and* M *is a bijection* $\phi \in \mathbb{P}((S_N \times S_M) \cup (T_N \times T_M))$ *such that for all* $d, e \in T_N \cup S_N$*:* $\ell_N(d) = \ell_M(\phi(d))$ *and* $F_N(d,e) = F_M(\phi(d), \phi(e))$*. LPT nets* N *and* M *are called* isomorphic *iff there exists an isomorphism between them.*

Clearly, isomorphism is an equivalence relation on LPT nets. Up to isomorphism, LPT nets are depicted by drawing the places as circles and the transitions as squares, with their labels inscribed. These elements are connected by arrows representing arcs, where the weight of the corresponding arc, if greater than 1, is shown. For reference, identifiers (not labels!) can be added next to the components (places or transitions) of nets.

Example The Figure 1 shows two LPT nets N and M. Both nets possess places with labels a and b and transitions with labels h and k. All the arcs from the a-labeled places in N and some arcs from the a-labeled places in M have weight 2. Clearly, N and M are not isomorphic. □

Process graph isomorphism is considered too fine an equivalence relation for concurrent systems as it allows hardly any verifications. It distinguishes a process x from the choice between x and x. Bisimulation abstracts from choices between equivalent alternatives while preserving choices that do affect the future behavior of a process. It is said that bisimulation preserves the *branching structure* of a process.

Likewise, we define a structural equivalence relation on LPT nets that is coarser than isomorphism but preserves the branching structure of the net, i.e. the way in which causes determine effects and the point where effects with the same cause diverge. Two LPT nets are considered equivalent iff their places and transitions can be related in such a way that (1) only elements of the same kind and with the same label are related, (2) causes and effects of related transitions are related, and (3-4) a bag related to a transition cause is a cause of some related transition.

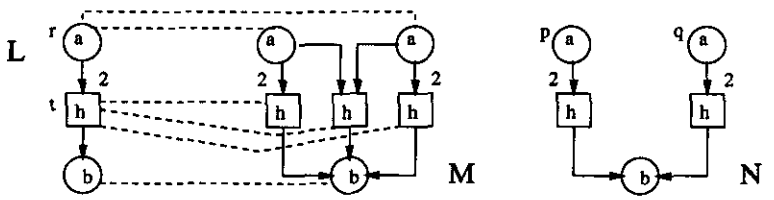

Figure 2: A bisimulation

Definition 3 *Let N, M be LPT nets. A* structural bisimulation *between N and M is a total and surjective relation $R \in \mathbb{P}((S_N \times S_M) \cup (T_N \times T_M))$ such that for all $t \in T_N$; $u \in T_M$; $d \in S_N \cup T_N$; $e \in S_M \cup T_M$; $P \in \mathbb{B}(S_N)$; $Q \in \mathbb{B}(S_M)$,*

1 $d\ R\ e \quad \Rightarrow \quad \ell_N(d) = \ell_M(e)$,
2 $t\ R\ u \quad \Rightarrow \quad C_N(t)\ \hat{R}\ C_M(u) \wedge E_N(t)\ \hat{R}\ E_M(u)$
3 $C_N(t)\ \hat{R}\ Q \quad \Rightarrow \quad \exists v \in T_M : C_M(v) = Q \wedge t\ R\ v$,
4 $P\ \hat{R}\ C_M(u) \quad \Rightarrow \quad \exists v \in T_N : C_N(v) = P \wedge v\ R\ u$,

Nets N and M, are called bisimilar *iff there exists a bisimulation between them. If there exists a* functional *bisimulation between N and M, M is called a* projection *of N. A total and surjective relation satisfying conditions 1 and 2 above is called* action preserving.

We simply use the term "bisimulation" in the context of LPT nets to denote structural bisimulation. Note that for state machines (all transitions having singleton cause and effect) with all places having the same label, structural bisimilarity coincides with "standard" bisimilarity (c.f. [BaVe95]). Also note the asymmetry in parts 3 and 4 of the definition. The absence of an analogous requirement for the E function signifies that bisimulation is biased toward the *future* of a marking, rather than its *history*.

Example The Figure 2 illustrates a bisimulation between the nets L and M. The bisimulation is indicated by dotted lines. Note that the nets L and N are not bisimilar. This can be deduced from the fact that the bag p^1q^1 in N must be related to the bag r^2 in L if a bisimulation existed. Now, $r^2 = C_L(t)$, whereas no transition u in N exists such that $C_N(u) = p^1q^1$. □

Lemma 1 *The following properties hold. Let N, M, L be LPT nets.*

i) An isomorphism between N and M is a bisimulation.

ii) The inverse of a bisimulation between N and M is a bisimulation between M and N.

iii) The composition of a bisimulation between N and M and a bisimulation between M and L is a bisimulation between N and L.

iv) Bisimilarity is an equivalence relation on LPT nets.

v) A bijective action preserving relation between N and M is an isomorphism.

vi) If N is a projection of M and M a projection of N, then N and M are isomorphic.

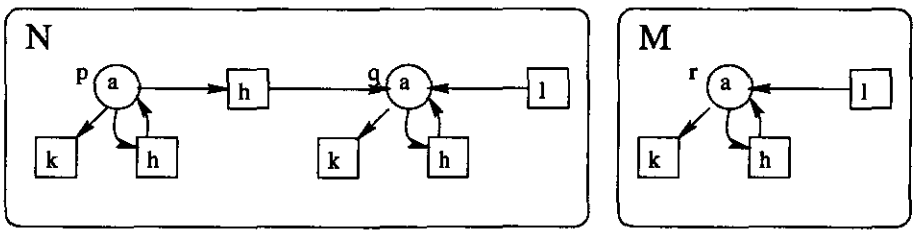

Figure 3: Bisimilar nets

Proof: The first three properties are trivial from the definition. The fourth follows from the first three. The identity isomorphism yields reflexivity. Taking the inverse yields symmetry. Composition yields transitivity. To prove the fifth, let ϕ be a bijective action preserving relation between N and M and let $t \in T_N$. By action preservation condition 2, $\hat{\phi}(E_N(t)) = E_M(\phi(t))$ and since ϕ is bijective, $E_N(t)(p) = \hat{\phi}(E_N(t))(\phi(p))$. So we deduce for any $p \in S_N$ that $F_N(t,p) = E_N(t)(p) = E_M(\phi(t))(\phi(p)) = F_M(\phi(t),\phi(p))$. Similarly, $F_N(p,t) = F_M(\phi(p),\phi(t))$. The last property follows from the fifth, since a projection that does not reduce the number of places and transitions of a net must be bijective. □

The following property illustrates the way in which structural properties of nets are preserved under bisimilarity.

Definition 4 *Let N be an LPT net and P a set of places of N.*
P is a siphon *of N iff* $\forall t \in T_N : \Sigma_{p \in P} F_N(t,p) > 0 \Rightarrow \Sigma_{p \in P} F_N(p,t) > 0$.
P is a trap *of N iff* $\forall t \in T_N : \Sigma_{p \in P} F_N(p,t) > 0 \Rightarrow \Sigma_{p \in P} F_N(t,p) > 0$.

Property 1 *If N, M are bisimilar LPT nets, then to each trap P of N there corresponds a trap Q of M having the same set of labels.*

Proof: Let R be a bisimulation between N and M and let P be a trap of N. We will show that the set $Q = \{q \in S_M \mid \exists p \in P : p\ R\ q\}$ is a trap of M. Suppose P is a trap and let $q \in Q$; $u \in T_M$ be such that $F_M(q,u) > 0$, so $q \in C_M(u)$. It suffices to show that there exists a $q' \in Q$ such that $q' \in E_M(u)$. By Definition 3.3, there exists a $p \in P$; $t \in T_N$, such that $p\ R\ q, t\ R\ u$ and $p \in C_N(t)$. Since P is a trap and $F_N(p,t) > 0$, there exists a $p' \in P$ such that $p' \in E_N(t)$. Thus by definition 3.2, there exists a q' such that $p'\ R\ q'$ (so $q' \in Q$) and $q' \in E_M(u)$. □

An analogous property for siphons does not exist, as is shown by Figure 3. The place p is a siphon of net N, but the bisimilar net M does not contain a siphon.

4 Algorithms

In this section, we study equivalence classes of LPT nets modulo bisimilarity. We use a standard technique (see e.g. [Cau90]) to arrive at a normal form for each equivalence class (modulo isomorphism).

We define a class of bisimulations of a given net N with itself called *congruences*. Unlike ordinary bisimulations, this class can be closed under (relation) union and we can compute the largest

congruence R of N w.r.t. set inclusion. We prove that R can be regarded as an equivalence relation upon the places and transitions of N and construct the "quotient" N/R. This quotient net is bisimilar to N and is unique up to isomorphism.

Computing the largest congruence of a net N is polynomial-bounded w.r.t. the number of places and transitions of N, but no effort has been made to optimize the straightforward algorithm.

Checking whether two LPT nets are isomorphic is equivalent to checking graph isomorphism. This even holds for the subclass of nets in normal form w.r.t. bisimilarity ([Bro96]). Graph isomorphism may be intractable (c.f. [GaJo79]), but in practice Petri net isomorphism is easily established, e.g. by the algorithm in [Mit88]. In the first step of this algorithm, the initial partitioning must also be done on the basis of label and kind (transition vs. place).

Definition 5 *A* congruence *of an LPT net N is a bisimulation R between N and N that satisfies for $d, e, f \in S_N \cup T_N$,*
reflexivity $d\ R\ d$
symmetry $d\ R\ e \Rightarrow e\ R\ d$
transitivity $d\ R\ e \wedge e\ R\ f \Rightarrow d\ R\ f$
An action preserving relation between N and N satisfying these conditions is called congruent.

We give a characterization for congruences that is easier to check than the defining conditions. In a congruence that relates places p and q, to every transition t with $p \in C(t)$ corresponds a transition u such that $C(u) = C(t) - p^1 + q^1$ and any congruent action preserving relation with this property is a congruence.

Lemma 2 *Let N be an LPT net. A congruent action preserving relation $R \in \mathbb{P}((S_N \times S_N) \cup (T_N \times T_N))$ is a congruence iff for all $p, q \in S_N$ and $t \in T_N$,*
$p\ R\ q \wedge p \in C_N(t) \Rightarrow \exists u \in T_N : t\ R\ u \wedge C_N(u) = C_N(t) - p^1 + q^1$.

Proof: Suppose that R is a congruence and let $p, q \in S_N$; $t \in T_N$ such that $p \in C_N(t)$ and $p\ R\ q$. Since $p\ R\ q$ and $p\ R\ p$, we have that $C_N(t)\ \hat{R}\ (C_N(t) - p^1 + q^1)$, so there must exist a $u \in T_N$ such that $t\ R\ u$.

Conversely, let R satisfy the lemma conditions and let $t \in T_N$; $Q \in \mathbb{B}(S_N)$ such that $C_N(t)\ \hat{R}\ Q$. There must exist a $k \geq 0$ and $p_1, q_1 \ldots, p_k, q_k$ such that $Q = C_N(t) - p_1^1 + q_1^1 \ldots - p_k^1 + q_k^1$. By the lemma conditions we conclude that there exist $u_1, \ldots, u_k \in T_N$ such that $t\ R\ u_1 \ldots\ R\ u_k$ and for $l \in 1..k$, $C_N(u_l) = C_N(t) - p_1^1 + q_1^1 \ldots - p_l^1 + q_l^1$, so $Q = C_N(u_k)$.

Since R is transitive, we conclude that $t\ R\ u_k$. By symmetry, $Q\ \hat{R}\ C_N(u)$ implies the existence of a $t \in T_N$ such that $Q = C_N(t)$ and $t\ R\ u$. So R is a bisimulation, and thus a congruence. □

As a corollary, we deduce that the transitive closure of the union of two congruences is again a congruence. This entails that every LPT net N possesses a maximal congruence, which is obtained by taking the transitive closure of the union of all congruences of N.

Lemma 3 *Let N be an LPT net and let R, S be congruences of N. Then $(R \cup S)^+$ is also a congruence of N.*

Proof: Clearly, the union of R and S is action preserving, reflexive and symmetric, so $(R \cup S)^+$ is congruent. We shall prove that $(R \cup S)^+$ satisfies the condition of the previous lemma. Let $p, q \in S_N$; $t \in T_N$ such that $p\ (R \cup S)^+\ q$ and $p \in C_N(t)$. By the definition of $(R \cup S)^+$, there must exist $p_1 \ldots, p_k \in S_N$ and $Q_1 \ldots, Q_k, Q_{k+1} \in \{R, S\}$ such that $p\ Q_1\ p_1 \ldots\ Q_k\ p_k\ Q_{k+1}\ q$. By the condition of the previous lemma, there must exist $t_1 \ldots, t_k, u \in T_N$ such that $C_N(t_1) = C_N(t) - p^1 + p_1^1 \ldots, C_N(t_k) = C_N(t_{k-1} - p_{k-1}^1 + p_k^1)$ and $C_N(u) = C_N(t_k) - p_k^1 + q^1$, whereas $t\ Q_1\ t_1 \ldots\ Q_k\ t_k\ Q_{k+1}\ u$. So $C_N(u) = C_N(t) - p^1 + q^1$ and $t\ (R \cup S)^+\ u$. □

Given an LPT net N and a congruence R of N, we construct the net $M = N/R$ (N modulo the congruence) as follows.

Construction 1 *Partition S_N and T_N into sets of elements related by R, so that elements from different sets are unrelated. For $d \in S_N \cup T_N$ let $\bar{d}$ denote the set containing d. Let $S_M = \{\bar{p} \mid p \in S_N\}$ and $T_M = \{\bar{t} \mid t \in T_N\}$ respectively. For $\bar{t} \in T_M$; $\bar{p} \in S_M$, set $F_M(\bar{t}, \bar{p}) = \Sigma_{q \in \bar{p}} F_N(t, q)$ and $F_M(\bar{p}, \bar{t}) = \Sigma_{q \in \bar{p}} F_N(q, t)$ for some $t \in \bar{t}$. Finally, let $\ell_M(\bar{d}) = \ell_N(d)$.*

Note that the construction of ℓ_M does not depend upon the choice of d. Let $t, u \in T_N$ with $\bar{t} = \bar{u}$, so $t\ R\ u$ and let $\bar{p} \in S_M$. Since R is action preserving, we have $C_N(t)\ \hat{R}\ C_N(u)$, so $\Sigma_{b \in \bar{p}} F_N(b, t) = \Sigma_{b \in \bar{p}} F_N(b, u)$. Thus $F_M(\bar{t}, \bar{p})$ does not depend upon the choice of t. The argument for $F_M(\bar{p}, \bar{t})$ is similar.

Example In Figure 2, the net M possesses a congruence R, which can be obtained by relating all nodes with the same label. The net L is isomorphic to M/R. □

Lemma 4 *Let N be an LPT net and let R be a congruence of N. The net N/R is bisimilar to N.*

Proof: Let $\rho = \{(d, \bar{d}) \mid d \in S_N \cup T_N\}$. We shall prove that ρ is a bisimulation between N and $M = N/R$. Note that ρ and thus $\hat{\rho}$ are functions, so for $d \in S_N \cup T_N$; $e \in S_M \cup T_M$; $D \in \mathbb{B}(S_N) \cup \mathbb{B}(T_N)$; $E \in \mathbb{B}(S_M) \cup \mathbb{B}(T_M)$ we have $\rho(d) = \rho(e) \Leftrightarrow d\ R\ e$ and $\hat{\rho}(D) = \hat{\rho}(E) \Leftrightarrow D\ \hat{R}\ E$.

Clearly, $\ell_N(d) = \ell_M(\bar{d})$, so bisimulation property 1 holds.

Let $t \in T_N$. We want to show $C_N(t)\ \hat{\rho}\ C_M(\bar{t})$, and thus $\hat{\rho}(C_N(t)) = C_M(\bar{t})$. Now $\hat{\rho}(C_N(t)) = \hat{\rho}(\Sigma_{p \in S_N} F_N(p, t).p^1) = \Sigma_{p \in S_N} \hat{\rho}(F_N(p, t).p^1) = \Sigma_{p \in S_N} F_N(p, t).\bar{p}^1 = \Sigma_{\bar{p} \in S_M} \Sigma_{q \in \bar{p}} F_N(t, q).\bar{q}^1 = \Sigma_{\bar{p} \in S_M}(\Sigma_{q \in \bar{p}} F_N(t, q)).\bar{p}^1 = \Sigma_{\bar{p} \in S_M} F_M(\bar{t}, \bar{p}).\bar{p}^1 = C_M(\bar{t})$. Similarly, $\hat{\rho}(E_N(t)) = E_M(\bar{t})$. This completes our check of bisimulation property 2.

Now let $t \in T_N$; $Q \in \mathbb{B}(S_M)$ such that $C_N(t)\ \hat{\rho}\ Q$. By the foregoing, $Q = \hat{\rho}(C_N(t)) = C_M(\bar{t})$, proving property 3.

Finally, let $\bar{u} \in T_M$; $P \in \mathbb{B}(S_N)$ such that $P\ \hat{\rho}\ C_M(\bar{u})$, so $\hat{\rho}(P) = C_M(\bar{u})$. Choose a u in $\bar{u}$. Then $\hat{\rho}(C_N(u)) = C_M(\bar{u})$ and $\rho(u) = \bar{u}$. Since $\hat{\rho}(P) = \hat{\rho}(C_N(u))$ we must have that $C_N(u)\ \hat{R}\ P$. Since R is a bisimulation, there must be a $t \in T_N$ such that $C_N(t) = P$ and $t\ R\ u$. Thus, $\bar{t} = \bar{u}$, so $t\ \rho\ \bar{u}$. This proves property 4 and thus completes our proof. □

The function $\rho = \{(d, \bar{d}) \mid d \in S_N \cup T_N\}$ is called the *canonical projection* of N onto N/R. Normalizing an LPT net N consists of finding the maximal congruence R within N.

Theorem 1 *Let R be a maximal congruence of an LPT net N. Then N/R is - up to isomorphism - a unique normal form of the equivalence class of LPT nets bisimilar to N.*

Given an LPT net N, compute a maximal congruence R of N.

$R \leftarrow \{p, q \in S_N \mid \ell(p) = \ell(q)\} \cup \{t, u \in T_N \mid \ell(t) = \ell(u)\}$,
$U \leftarrow$ **true**;
while U do
  $U \leftarrow$ **false**;
  for $(d, e) \in R$ do
    if $d \in S_N \wedge (\exists t \in \mathrm{dom}(E_N(d))$:
      $(\forall u \in T_N : t\ R\ u \Rightarrow C_N(u) \neq C_N(t - d^1 + e^1))$
    $\vee d \in T_N \wedge (C_N(d), C_N(e)) \notin \hat{R} \vee (E_N(d), E_N(e)) \notin \hat{R})$
    then $R \leftarrow R - \{(d, e)\}$, $U \leftarrow$ **true** fi
  od
od.

Table 1: Maximal congruence algorithm

Proof: Let ρ be the bisimulation between N and N/R. If Q is a bisimulation between a net M and N, the transitive closure of $Q^{-1} \circ Q$ is a congruence of N, and thus must be contained in R (by the maximality). So for $d, e \in S_N \cup T_N$; $f \in S_M \cup T_M$ we have $f\ Q\ d \wedge f\ Q\ e \Rightarrow d\ R\ e$. So $Q \circ \rho$ is a projection. Thus, by Lemma 1.6, N/R is unique up to isomorphism. □

The algorithm in Table 1 constructs the maximal congruence of a given net N. Initially all elements with the same label are related to one another. Any related pair violating the conditions of a bisimulation is removed. The removal of relations may cause other pairs to violate the conditions. This process is repeated until no pair needs to be removed anymore. The resulting relation is the maximal congruence of the net. The boolean variable U indicates whether an update of the relation R has occurred in the current loop.

The algorithm gives a congruence, for if U remains **false**, all pairs in R have been checked to satisfy the condition of Lemma 2 as well as the action preservation condition. It contains the identity relation, so it is reflexive. The loop invariant is that R contains the maximal reflexive bisimulation as a subrelation, which proves that R is maximal. It must therefore be symmetric and transitive and thus a congruence.

5 Process theory

We now define *processes* associated to LPT nets. We start by defining transition systems augmented with a partial order, and where states, like steps, are labeled. Let L be a fixed set of labels.

Definition 6 *An* ordered process space *consists of disjoint sets P of states, and A of steps together with a ternary step (transition) relation $\_ \dot{\longrightarrow} \_$ in $\mathbb{P}(P \times A \times P)$, partial orders $\leq$ on A and on P and a function ℓ in $(A \cup P) \rightarrow L$.*

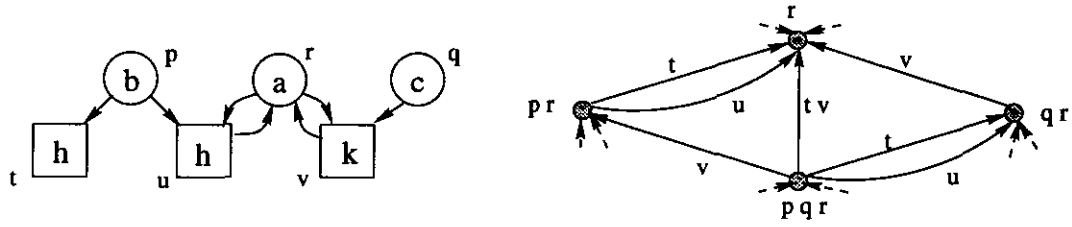

Figure 4: Net and process space

For ordered process spaces, bisimulation equivalence can be extended with order preservation. Ordered process spaces are order preserving bisimilar iff their states and steps can be related in such a way that related objects are of the same kind, have the same label and have the "transfer property" w.r.t. both the transition relation and partial order.

Definition 7 *Let V, W be ordered process spaces. A total and surjective relation $R \in \mathbb{P}((P_V \times P_W) \cup (A_V \times A_W))$ is called an order preserving bisimulation (or OP-bisimulation for short) between V and W iff for all $d, d', d'' \in P_V \cup A_V$; $e, e', e'' \in P_W \cup A_W$ such that $d\ R\ e$,*

$$
\begin{array}{lll}
1 & \ell_V(d) = \ell_W(e), & \\
2 & d \leq_V d' & \Rightarrow\ \exists f \in P_W \cup A_W : e \leq_W f \wedge d'\ R\ f, \\
3 & e \leq_W e' & \Rightarrow\ \exists f \in P_V \cup A_V : d \leq_V f \wedge f\ R\ e', \\
4 & d \xrightarrow{d'}_V d'' & \Rightarrow\ \exists f \in A_W, g \in P_W : e \xrightarrow{f}_W g \wedge d'\ R\ f \wedge d''\ R\ g, \\
5 & e \xrightarrow{e'}_W e'' & \Rightarrow\ \exists f \in A_V, g \in P_V : d \xrightarrow{f}_V g \wedge f\ R\ e' \wedge g\ R\ e'', \\
6 & d' \xrightarrow{d}_V d'' & \Rightarrow\ \exists f, g \in P_W : f \xrightarrow{e}_W g \wedge d'\ R\ f \wedge d''\ R\ g, \\
7 & e' \xrightarrow{e}_W e'' & \Rightarrow\ \exists f, g \in P_V : f \xrightarrow{d}_V g \wedge f\ R\ e' \wedge g\ R\ e''.
\end{array}
$$

V and W are called OP-bisimilar *iff there exists an OP-bisimulation R between them.*

Bisimilarity is an equivalence relation, which can be proved by taking the identity relation for reflexivity, the inverse relation for symmetry and relation composition for transitivity. Note that if the partial order is trivial (i.e. $d \leq e$ either always or only if $d = e$) and all states have the same label, steps with the same label can always be related and OP-bisimilarity corresponds to "standard" bisimilarity.

An LPT net N defines an ordered process space V by defining states as *markings* or bags of places and steps as bags of transitions, the partial order by bag inclusion and the transition relation by token consumption and production, allowing for autoconcurrency (firing of the same transition concurrently with itself).

Definition 8 *Let $L = \mathbb{B}(\mathcal{L})$. An LPT net N defines an ordered process space $V = V_N$ as follows. $P_V = \mathbb{B}(S_N)$; $A_V = \mathbb{B}(T_N)$; $\ell_V = \hat{\ell}_N$. The partial order $\leq$ on V is given by bag inclusion. For $p, p' \in P_V$; $\alpha \in A_V$ we have $p \xrightarrow{\alpha}_V p'$ iff $(\Sigma_{t \in \alpha} C_N(t)) \leq p \wedge p' = p - (\Sigma_{t \in \alpha} C_N(t)) + (\Sigma_{t \in \alpha} E_N(t))$.*

Example The Figure 4 shows a net and part of the process space defined by it. The net is in normal form w.r.t. structural bisimulation and thus cannot be reduced. This net can be interpreted as a

factory floor containing three machines. The machines t, u can process the same jobs b, whereas the machine v processes jobs c. The machines u, v need an a-resource (operator). The t machine can work without operator.

In the process space, the 1-superscripts of the states and steps have been omitted. Note that the steps t^1 and u^1 cannot be related by an OP-bisimulation, since $t^1 \leq t^1 v^1$ and there does not exist a step $\geq u^1$ between the states $p^1 q^1 r^1$ and r^1. The interpretation is that the machine u can occupy a resource also needed by v, so they cannot work concurrently. This behavior does not occur when u is removed, so the factory floor with and without u have a different (qualitative!) behavior. □

We now prove our main theorem relating structurally bisimilar nets to order preserving bisimilar process spaces.

Theorem 2 *Let N, M be LPT nets and let V, W be the respective process spaces defined by them. Then V and W are OP-bisimilar iff N and M are structurally bisimilar.*

Proof: We first prove the "if" part. Let R be a structural bisimulation between N and M. Let Q be the relation between V and W defined by $Q = \hat{R}$. We shall prove that Q is an OP-bisimulation. Bisimulation conditions 1-3 for Q can be verified trivially.

We prove condition 4. Condition 5 then follows by symmetry. Let $d' \in A_V; d, d'' \in P_V; e \in P_W$ such that $d \; Q \; e$ and $d \xrightarrow{d'}_V d''$. Then $(\Sigma_{t \in d'} C_N(t)) \leq d$ and $d'' = d - (\Sigma_{t \in d'} C_N(t)) + (\Sigma_{t \in d'} E_N(t))$. For $t \in d'$, we can find bags $B_t \in P_W$ such that $\Sigma_{t \in d'} B_t \leq e$ and $\forall t \in d' : C_N(t) \; Q \; B_t$. Since R is a bisimulation between N and M, there must exist a $u_t \in T_M$ such that $C_N(u_t) = B_t$ and $t \; R \; u_t$. Take $e' = \Sigma_{t \in d'} u_t^1$. Clearly, $d' \; Q \; e'$. Moreover, $\Sigma_{u \in e'} C_N(u) = \Sigma_{t \in d'} C_N(u_t) = \Sigma_{t \in d'} B_t \leq e$. So there exists a (unique) e'' such that $e \xrightarrow{e'}_W e''$ Since $d' \; \hat{R} \; e'$, there exists a 1-1 correspondence between the t in d' and u in e' such that corresponding t and u satisfy $t \; R \; u$. Since R is action preserving, it follows that $\Sigma_{t \in d'} E_N(t) \; \hat{R} \; \Sigma_{u \in e'} E_N(u)$. By the definition of d'' and e'', we conclude that $d'' \; Q \; e''$.

We prove condition 6. Condition 7 then follows by symmetry. Let $d \in A_V; d', d'' \in P_V; e \in A_V$ such that $d \; Q \; e$ and $d' \xrightarrow{d}_V d''$. Let $d_0 = \Sigma_{t \in d} C_N(t)$. Then $d_0 \leq d'$ and $d'' = d' - d_0 + \Sigma_{t \in d} E_N(t)$. For each $t \in d$ we can find a $u_t \in e$ such that $t \; R \; u_t$ and $e = \Sigma_{t \in d} u_t^1$. Let $e_0 = \Sigma_{t \in d} C_N(u_t)$. Since $d_0 \; Q \; e_0$, we can find an $e_1 \in P_W$ such that $d' \; Q \; e_0 + e_1$. Take $e' = e_0 + e_1$ and e'' such that $e' \xrightarrow{e}_W e''$. Clearly, $d'' \; Q \; e''$ as well. This settles the "if" part.

We now prove the "only if" part. Let Q be an OP-bisimulation between V and W. Note that, by OP-bisimulation condition 1, we can deduce that in this case OP-bisimulation conditions 2-3 also hold when strict inclusion is substituted for inclusion. So from Lemma 5 we may deduce that there exists a total and surjective relation $R \in \mathbb{P}((S_N \times T_N) \cup (S_M \times T_M))$ such that $Q = \hat{R}$. We shall prove that R is a structural bisimulation between N and M. Property 1 is easy.

We prove net bisimulation property 2. Let $t \in T_N$ and $u \in T_M$ such that $t \; R \; u$. We know that $C_N(t) \xrightarrow{t^1}_V E_N(t)$ and $C_N(u) \xrightarrow{u^1}_W E_N(u)$, whereas $t^1 \; Q \; u^1$. Since Q is an OP-bisimulation, we deduce that $C_N(t) \; Q \; C_N(u)$ and $E_N(t) \; Q \; E_N(u)$.

We prove net bisimulation property 3. Property 4 then follows by symmetry. Let $t \in T_N; B \in \mathbb{B}(S_M)$ such that $C_N(t) \; Q \; B$. Since $C_N(t) \xrightarrow{t^1}_V E_N(t)$, there must exist $\beta \in A_W$, $B' \in P_W$ such

that $B \xrightarrow{\beta}_W B'$ and $t^1\ Q\ \beta$. Since t^1 is a singleton, β must be a singleton, say u^1 and $t\ R\ u$. Moreover, $C_N(t)$ is the smallest state in P_V admitting the step t^1. It follows from properties 2-3 of Q that B has the same property w.r.t. the step u. So $B = C_N(u)$. Thus R is indeed a structural bisimulation. □

Lemma 5 *Let A, B be finite sets and let $Q \in \mathbb{P}(\mathbb{B}(A) \times \mathbb{B}(B))$ be a total and surjective relation satisfying for $\alpha, \alpha' \in \mathbb{B}(A)$; $\beta, \beta' \in \mathbb{B}(B)$ such that $\alpha\ Q\ \beta$,*

$\alpha < \alpha' \ \Rightarrow\ \exists \gamma \in \mathbb{B}(B) : \beta < \gamma \wedge \alpha'\ Q\ \gamma$,

$\beta < \beta' \ \Rightarrow\ \exists \gamma \in \mathbb{B}(A) : \alpha < \gamma \wedge \gamma\ Q\ \beta'$.

Then there exists a total and surjective relation $R \in \mathbb{P}(A \times B)$ such that $Q = \hat{R}$.

Proof: Note that all bags are finite since A and B are finite. We prove by induction that Q equals some $\hat{R}$ when restricted to bags of size $\leq k$. Because of the conditions on Q and since $\mathbf{0}$ is the minimal element in both sets, we deduce that Q relates $\mathbf{0}$ to and only to $\mathbf{0}$. Since singleton bags are the only bags that have $\mathbf{0}$ as a strict subbag, Q must relate singletons to and only to singletons. This settles the cases for sizes 0 and 1. Clearly, R must be the relation defined by $a\ R\ b \Leftrightarrow a^1\ Q\ b^1$. Let $\alpha \in \mathbb{B}(A)$; $\beta \in \mathbb{B}(B)$ such that $\alpha\ Q\ \beta$. Let α_i for $i \in I$ be the set of all strict subbags of α and let β_j for $j \in J$ be the set of all strict subbags of β. By the conditions on Q, each α_i must be Q-related (and thus $\hat{R}$-related by the induction hypothesis) to a β_j and vice versa. By some case analysis it is easy to show that $\alpha\ \hat{R}\ \beta$. □

6 Behavioral properties of nets

Theorem 2 shows how behavioral properties of LPT nets are preserved under structural bisimulation. We define a few of these properties. If N is an LPT net and V the process space derived from it, we adopt the notation $P\ [A\rangle_N\ Q$ for $P \xrightarrow{A}_V Q$.

Definition 9 *A* marked LPT net *is a pair (N, P) where N is an LPT net and $P \in \mathbb{B}(S_N)$ a marking of N. A marking $Q \in \mathbb{B}(S_N)$ is called* reachable *from (N, P) (notation $P\ [*\rangle_N\ Q$) iff there exist $k > 0$; $P = Q_1, \ldots, Q_k = Q \in \mathbb{B}(S_N)$; $A_1 \ldots A_{k-1} \in \mathbb{B}(T_N)$ such that $Q_i\ [A_i\rangle_N\ Q_{i+1}$ for $0 < i < k$. The marked net (N, P) is called* bounded *iff there exists an $n > 0$ such that every $Q \in \mathbb{B}(S_N)$ with $P\ [*\rangle_N\ Q$ has size $< n$. It is* live *iff for every $Q \in \mathbb{B}(S_N)$ with $P\ [*\rangle_N\ Q$ and $t \in T_N$ there exists a Q', $Q'' \in \mathbb{B}(S_N)$ such that $Q\ [*\rangle_N\ Q'\ [t^1\rangle_N\ Q''$. It* deadlocks *iff there is no $t \in T_N$; $Q \in \mathbb{B}(S_N)$ such that $P\ [t^1\rangle_N\ Q$. It* has a deadlock *iff there exists a $Q \in \mathbb{B}(S_N)$ with $P\ [*\rangle_N\ Q$ such that Q deadlocks.*

The proof of the following property is trivial from Theorem 2. Note that liveness is not completely preserved. A counterexample can be found in Figure 3. This counterexample - and the whole idea behind transition labeling - suggest a weakened definition of liveness: an LPT net is live iff for every reachable state Q and transition t a subsequent state Q' and transition u *with the same label* can be found such that u can fire.

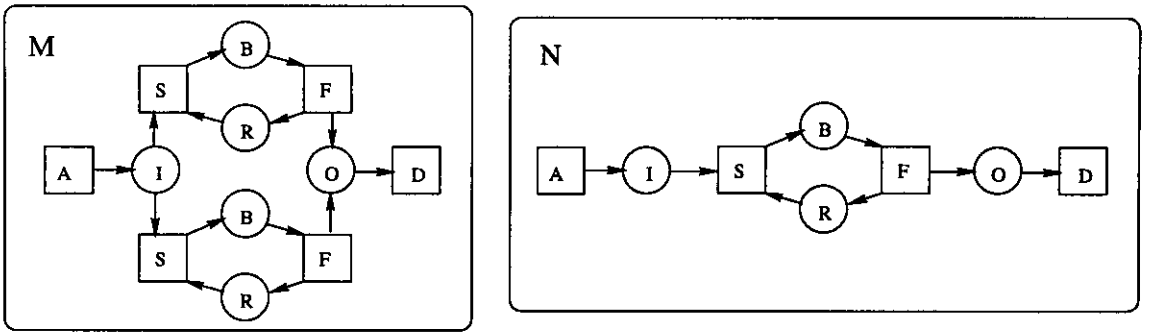

Figure 5: Queuing systems

Property 2 *Let (N, P) be a marked LPT net, R a congruence of N, $M = N/R$, ρ the canonical projection of N to M and $Q = \rho(P)$. Then*
(N, P) is bounded iff (M, Q) is bounded,
(N, P) deadlocks iff (M, Q) deadlocks,
(N, P) has a deadlock iff (M, Q) has a deadlock,
(N, P) is live implies (M, Q) is live.

7 Example

In Figure 5 two bisimilar nets are depicted that implement queues. The I(input)-labeled place contains clients that are to be served by the system. Tokens are produced for this place by the firing of the A(arrival) transition. The R(resource) places contain available servers. When an S(start) transition fires, a client and server are consumed and a B(busy) client-server combination is produced. When F(finish) fires, this B combination is consumed, the client is transferred to O(output) and the server to R. The D(departure) transition can remove these clients.

In M, two parallel queue systems are depicted, whereas N is a single queue system. Clearly, both nets are structurally bisimilar, so their process spaces are OP-bisimilar. This means that e.g. adding two tokens in place R of N and one each in the resource places of M will result in processes that behave the same.

Interleaving bisimilarity of the "action" parts of the nets N and M can also be proved by more or less standard techniques, viz. BPA_δ with recursion (c.f. [BaVe95]). In this case, steps consist of the firing of a single transition and states are not labeled. The nodes in the process space V_N correspond to markings of the net N. Let $X_{k,l,m,n}$ denote the node corresponding to the marking $I^k B^l R^m D^n$. We derive the following recursive equations (*) for $k, l, m, n \in \mathbb{N}$:

$$X_{k,l,m,n} = A.X_{k+1,l,m,n} + S_{k>0 \wedge m>0}.X_{k-1,l+1,m-1,n} + F_{l>0}.X_{k,l-1,m+1,n} + D_{n>0}.X_{k,l,m,n-1},$$

where A_P denotes the action A if P is true and δ if P is false. For the net M similar equations can be derived for nodes $Y_{k,l_1,l_2,m_1,m_2,n}$. From these equations it can be deduced that $Y_{k,l_1,l_2,m_1,m_2,n}$ and $X_{k,l_1+l_2,m_1+m_2,n}$ satisfy the same set of (guarded) equations derived from (*). The RSP rule then yields bisimilarity of the process spaces, giving another structural equivalence of the nets.

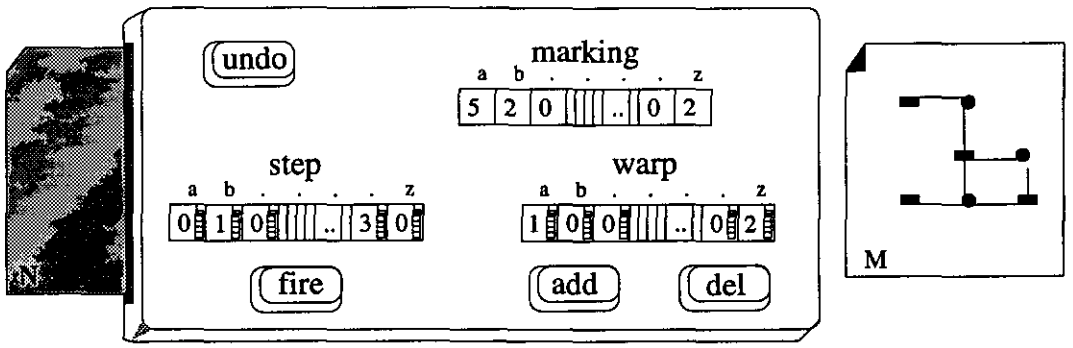

Figure 6: LPT net testing device

The method sketched above can be generalized to arbitrary finite nets, showing e.g. that the net in Figure 4 and the same net with u deleted are bisimilar. It has the advantage of an axiomatic approach, but is hard to derive algorithmically. The method of this paper has the advantage of simplicity, modeling states and a concurrent step semantics.

8 Conclusions and further work

One may ask what structural LPT net bisimilarity amounts to in practice. This question may be answered by a metaphor in the style of [Gla90]. An observer is given an LPT net M. At the same time, an LPT net N is inserted in a testing device, which is depicted in Figure 6.

The device possesses four buttons, labeled "fire", "add", "del" and "undo". It also possesses three rows of dials, labeled "marking", "step" and "warp". Each dial is labeled with a label from $\mathcal{L}$ and displays a nonnegative integer. The "step" and "warp" dials can be set by the observer.

Initially, the marking dials display zeros. To test the net N, the observer may set the "warp" dials and press "add". The device will respond by adding tokens to the net N. If the "warp" dial labeled with a is set with a number $n > 0$, then n tokens will be inserted in a-labeled places and likewise for the other labels. Tokens can be removed by setting the "warp" dial and pressing "del". If the "warp" dial labeled with a is set with a number $n > 0$, then n tokens will be removed from a-labeled places and likewise for the other labels. If the specified warp cannot be executed (there are no a-labeled places in N or these places do not contain enough tokens), the maximum possible number of tokens is added or removed. If there are several possibilities for adding or removing tokens, one of them is chosen nondeterministically. After the warp, the "warp" dials display the actual number of tokens added or removed and the "marking" dials display the marking (number of tokens in a-labeled places for each label a).

The observer may also set the "step" dials and press "fire". The device will respond by executing a step corresponding to the set dials. If the "step" dial labeled with h is set with a number $n > 0$, then n concurrent firings of h-labeled transitions will be included in the step and likewise for the other labels. If the specified step cannot be executed (e.g. there are not enough tokens available), a maximal substep of the specified step is chosen. From the possible steps, one is chosen nonde-

terministically.

The device, while executing a step F, displays in the "step" dials the number of concurrent firings of h-labeled transitions for each label h in F and displays in the "marking" dials first the state reached after consuming the tokens required for F and then the state reached after completing the step F.

At any time, the observer may press "undo", restoring the situation before his last action. The observer is assigned the task of experimenting with N through the device and trying to match his observations to the behavior of net M. He must report whether he can detect any differences between M and N.

Such a difference may arise when N displays an observable behavior that cannot be matched by any behavior of M. If N and M would be the nets in Figure 2, he could warp to the state a^2, and discover that firing h^1 blocks. Since such a deadlock cannot occur in M, he can report their being different.

However, had N and M been interchanged, he cannot be sure so easily. He can warp to a^2 and fire h^1 as often as he likes, but, for all he knows, every time he warps to a^2, a non-deadlocking state may be chosen. So the non-occurrence of a deadlock in N gives no information as to M and N being different. For this very reason the LPT net testing device comes with a "fairness warranty" from its manufacturer:

> By performing sufficiently many experiments with our device, every possible outcome will occur.

In small print, it is defined what "sufficiently many" means in terms of the size of the net N inserted and the nature of the experiments conducted. So after a while, failing to observe a deadlock, the observer may conclude that M and N are indeed different, provided that the number of places and transitions of the net N does not exceed a given upper bound.

We assert that LPT nets N and M are structurally bisimilar iff the observer cannot discover any differences between them in the above way. For instance, the difference between the net in Fig 4 and the same net with transition u removed can be discovered by warping to $a^1b^1c^1$ and discovering that one net sometimes refuses the step h^1k^1 and the other not.

The testing device described above is very powerful indeed. One may remove the fairness warranty (giving ready simulation), the undo button (giving failure-like equivalences) and/or the warp buttons and dials (giving behavioral equivalences). These possibilities certainly merit further investigation, especially the structural ones (allowing warps), as the behavioral ones will be similar to those mentioned in [PoRS92].

We have introduced structural net bisimulation and studied a few of its properties. Other properties of this equivalence relation, like causality preservation or congruence w.r.t. operations like refining transitions and/or places need further investigation.

The bisimilarity relations in this paper can be compared to *strong* bisimilarity for process graphs. Each token within the state and each transition firing in a step is of importance. In contrast, *weak* bisimilarity allows places and transitions to be labeled with the *invisible* label τ. Tokens in τ-labeled places should only be noticed by the fact that they enable steps that can be noticed. Sim-

ilarly, the inclusion of τ-labeled transitions in a step is only noticed by the fact that tokens are consumed or produced that can be noticed.

For practical verifications, some form of weak bisimilarity cannot be dispensed with, so an extension of the present theory in this direction is necessary. Another interesting extension is to high-level (e.g. colored) nets.

References

BaBe95 J.C.M. Baeten, J.A. Bergstra: *Process Algebra with Propositional Signals*, in: Ponse, Verhoef, Vlijmen (eds.) *Proc. ACP'95*, pp. 213-227, Computing Science Report 95-14 Eindhoven Univ. Tech. 1995.

BaVe95 J.C.M. Baeten, C. Verhoef: *Concrete Process Algebra*, in: Abramsky, Gabbay, Maibaum (eds.) *Handb. Logic Comp. Sci.*, Vol. 4, pp. 149-268, Oxford Univ. Press 1995.

BaVo95 T. Basten, M. Voorhoeve: *An Algebraic Semantics for Hierarchical P/T Nets (extended abstract)*, in: De Michelis, Diaz (eds.): *Proc. Appl. Theor. Petri Nets*, pp. 45-65, LNCS 935, Springer 1995.

BeDH92 E. Best, R. Devillers, J.G. Hall: *The Box Calculus: A New Causal Algebra with Multi-label Communication*, in: [Roz92], pp. 21-69.

Bro96 A.E. Brouwer (private communication)

Cau90 D. Caucal: *Graphes Canoniques de Graphes Algébriques*, RAIRO Inf. Theor. Appl. 24, pp. 339-352

GaJo79 M.R. Garey, D.S. Johnson: *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman 1979

Gla90 R.J. van Glabbeek: *The Linear Time - Branching Time Spectrum*, in: Baeten, Klop (eds.) *Proc. CONCUR 90*, pp. 278-297, LNCS 458, Springer 1990

Gla95 R.J. van Glabbeek: *History Preserving Process Graphs*, preprint.

GlVa87 R.J. van Glabbeek, F.W. Vaandrager: *Petri Net Models for Algebraic Theories of Concurrency*, in: de Bakker et al. (eds.) *Proc. PARLE, Vol. II: Parallel Languages*, pp. 224-242, LNCS 259, Springer 1987.

HeSV91 K.M. van Hee, L.J. Somers and M. Voorhoeve: *A Formal Framework for Dynamic Modelling of Information Systems* in: Sol, van Hee (eds.) *Dyn. Mod. Inf. Sys.*, pp. 227-236, North Holland 1991.

Jen92 K. Jensen: *Coloured Petri Nets, Vol. I: Basic Concepts*, EATCS monographs 28, Springer 1992.

Mit88 H.B. Mittal: *A Fast Backtrack Algorithm for Graph Isomorphism*, Inf. Proc. Letters 29 (1988), pp. 105-110.

PoRS92 L. Pomello, G. Rozenberg, C. Simone: *A Survey of Equivalence Notions for Net Based Systems*, in: [Roz92], pp. 410-472.

Roz92 G. Rozenberg (ed.): *Adv. Petri Nets 1992*, LNCS 609, Springer 1992

Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

93/01	R. van Geldrop	Deriving the Aho-Corasick algorithms: a case study into the synergy of programming methods, p. 36.
93/02	T. Verhoeff	A continuous version of the Prisoner's Dilemma, p. 17
93/03	T. Verhoeff	Quicksort for linked lists, p. 8.
93/04	E.H.L. Aarts J.H.M. Korst P.J. Zwietering	Deterministic and randomized local search, p. 78.
93/05	J.C.M. Baeten C. Verhoef	A congruence theorem for structured operational semantics with predicates, p. 18.
93/06	J.P. Veltkamp	On the unavoidability of metastable behaviour, p. 29
93/07	P.D. Moerland	Exercises in Multiprogramming, p. 97
93/08	J. Verhoosel	A Formal Deterministic Scheduling Model for Hard Real-Time Executions in DEDOS, p. 32.
93/09	K.M. van Hee	Systems Engineering: a Formal Approach Part I: System Concepts, p. 72.
93/10	K.M. van Hee	Systems Engineering: a Formal Approach Part II: Frameworks, p. 44.
93/11	K.M. van Hee	Systems Engineering: a Formal Approach Part III: Modeling Methods, p. 101.
93/12	K.M. van Hee	Systems Engineering: a Formal Approach Part IV: Analysis Methods, p. 63.
93/13	K.M. van Hee	Systems Engineering: a Formal Approach Part V: Specification Language, p. 89.
93/14	J.C.M. Baeten J.A. Bergstra	On Sequential Composition, Action Prefixes and Process Prefix, p. 21.
93/15	J.C.M. Baeten J.A. Bergstra R.N. Bol	A Real-Time Process Logic, p. 31.
93/16	H. Schepers J. Hooman	A Trace-Based Compositional Proof Theory for Fault Tolerant Distributed Systems, p. 27
93/17	D. Alstein P. van der Stok	Hard Real-Time Reliable Multicast in the DEDOS system, p. 19.
93/18	C. Verhoef	A congruence theorem for structured operational semantics with predicates and negative premises, p. 22.
93/19	G-J. Houben	The Design of an Online Help Facility for ExSpect, p.21.
93/20	F.S. de Boer	A Process Algebra of Concurrent Constraint Programming, p. 15.
93/21	M. Codish D. Dams G. Filé M. Bruynooghe	Freeness Analysis for Logic Programs - And Correctness, p. 24
93/22	E. Poll	A Typechecker for Bijective Pure Type Systems, p. 28.
93/23	E. de Kogel	Relational Algebra and Equational Proofs, p. 23.
93/24	E. Poll and Paula Severi	Pure Type Systems with Definitions, p. 38.
93/25	H. Schepers and R. Gerth	A Compositional Proof Theory for Fault Tolerant Real-Time Distributed Systems, p. 31.
93/26	W.M.P. van der Aalst	Multi-dimensional Petri nets, p. 25.
93/27	T. Kloks and D. Kratsch	Finding all minimal separators of a graph, p. 11.
93/28	F. Kamareddine and R. Nederpelt	A Semantics for a fine λ-calculus with de Bruijn indices, p. 49.
93/29	R. Post and P. De Bra	GOLD, a Graph Oriented Language for Databases, p. 42.
93/30	J. Deogun T. Kloks D. Kratsch H. Müller	On Vertex Ranking for Permutation and Other Graphs, p. 11.

93/31	W. Körver	Derivation of delay insensitive and speed independent CMOS circuits, using directed commands and production rule sets, p. 40.
93/32	H. ten Eikelder and H. van Geldrop	On the Correctness of some Algorithms to generate Finite Automata for Regular Expressions, p. 17.
93/33	L. Loyens and J. Moonen	ILIAS, a sequential language for parallel matrix computations, p. 20.
93/34	J.C.M. Baeten and J.A. Bergstra	Real Time Process Algebra with Infinitesimals, p.39.
93/35	W. Ferrer and P. Severi	Abstract Reduction and Topology, p. 28.
93/36	J.C.M. Baeten and J.A. Bergstra	Non Interleaving Process Algebra, p. 17.
93/37	J. Brunekreef J-P. Katoen R. Koymans S. Mauw	Design and Analysis of Dynamic Leader Election Protocols *in Broadcast Networks*, p. 73.
93/38	C. Verhoef	A general conservative extension theorem in process algebra, p. 17.
93/39	W.P.M. Nuijten E.H.L. Aarts D.A.A. van Erp Taalman Kip K.M. van Hee	Job Shop Scheduling by Constraint Satisfaction, p. 22.
93/40	P.D.V. van der Stok M.M.M.P.J. Claessen D. Alstein	A Hierarchical Membership Protocol for Synchronous Distributed Systems, p. 43.
93/41	A. Bijlsma	Temporal operators viewed as predicate transformers, p. 11.
93/42	P.M.P. Rambags	Automatic Verification of Regular Protocols in P/T Nets, p. 23.
93/43	B.W. Watson	A taxomomy of finite automata construction algorithms, p. 87.
93/44	B.W. Watson	A taxonomy of finite automata minimization algorithms, p. 23.
93/45	E.J. Luit J.M.M. Martin	A precise clock synchronization protocol,p.
93/46	T. Kloks D. Kratsch J. Spinrad	Treewidth and Patwidth of Cocomparability graphs of Bounded Dimension, p. 14.
93/47	W. v.d. Aalst P. De Bra G.J. Houben Y. Komatzky	Browsing Semantics in the "Tower" Model, p. 19.
93/48	R. Gerth	Verifying Sequentially Consistent Memory using Interface Refinement, p. 20.
94/01	P. America M. van der Kammen R.P. Nederpelt O.S. van Roosmalen H.C.M. de Swart	The object-oriented paradigm, p. 28.
94/02	F. Kamareddine R.P. Nederpelt	Canonical typing and Π-conversion, p. 51.
94/03	L.B. Hartman K.M. van Hee	Application of Marcov Decision Processe to Search Problems, p. 21.
94/04	J.C.M. Baeten J.A. Bergstra	Graph Isomorphism Models for Non Interleaving Process Algebra, p. 18.
94/05	P. Zhou J. Hooman	Formal Specification and Compositional Verification of an Atomic Broadcast Protocol, p. 22.
94/06	T. Basten T. Kunz J. Black M. Coffin D. Taylor	Time and the Order of Abstract Events in Distributed Computations, p. 29.
94/07	K.R. Apt R. Bol	Logic Programming and Negation: A Survey, p. 62.
94/08	O.S. van Roosmalen	A Hierarchical Diagrammatic Representation of Class Structure, p. 22.
94/09	J.C.M. Baeten J.A. Bergstra	Process Algebra with Partial Choice, p. 16.

94/10	T. verhoeff	The testing Paradigm Applied to Network Structure. p. 31.
94/11	J. Peleska C. Huizing C. Petersohn	A Comparison of Ward & Mellor's Transformation Schema with State- & Activitycharts, p. 30.
94/12	T. Kloks D. Kratsch H. Müller	Dominoes, p. 14.
94/13	R. Seljée	A New Method for Integrity Constraint checking in Deductive Databases, p. 34.
94/14	W. Peremans	Ups and Downs of Type Theory, p. 9.
94/15	R.J.M. Vaessens E.H.L. Aarts J.K. Lenstra	Job Shop Scheduling by Local Search, p. 21.
94/16	R.C. Backhouse H. Doombos	Mathematical Induction Made Calculational, p. 36.
94/17	S. Mauw M.A. Reniers	An Algebraic Semantics of Basic Message Sequence Charts, p. 9.
94/18	F. Kamareddine R. Nederpelt	Refining Reduction in the Lambda Calculus, p. 15.
94/19	B.W. Watson	The performance of single-keyword and multiple-keyword pattern matching algorithms, p. 46.
94/20	R. Bloo F. Kamareddine R. Nederpelt	Beyond β-Reduction in Church's $\lambda\rightarrow$, p. 22.
94/21	B.W. Watson	An introduction to the Fire engine: A C++ toolkit for Finite automata and Regular Expressions.
94/22	B.W. Watson	The design and implementation of the FIRE engine: A C++ toolkit for Finite automata and regular Expressions.
94/23	S. Mauw and M.A. Reniers	An algebraic semantics of Message Sequence Charts, p. 43.
94/24	D. Dams O. Grumberg R. Gerth	Abstract Interpretation of Reactive Systems: Abstractions Preserving $\forall$CTL*, $\exists$CTL* and CTL*, p. 28.
94/25	T. Kloks	$K_{1,3}$-free and W_4-free graphs, p. 10.
94/26	R.R. Hoogerwoord	On the foundations of functional programming: a programmer's point of view, p. 54.
94/27	S. Mauw and H. Mulder	Regularity of BPA-Systems is Decidable, p. 14.
94/28	C.W.A.M. van Overveld M. Verhoeven	Stars or Stripes: a comparative study of finite and transfinite techniques for surface modelling, p. 20.
94/29	J. Hooman	Correctness of Real Time Systems by Construction, p. 22.
94/30	J.C.M. Baeten J.A. Bergstra Gh. Ştefanescu	Process Algebra with Feedback, p. 22.
94/31	B.W. Watson R.E. Watson	A Boyer-Moore type algorithm for regular expression pattern matching, p. 22.
94/32	J.J. Vereijken	Fischer's Protocol in Timed Process Algebra, p. 38.
94/33	T. Laan	A formalization of the Ramified Type Theory, p.40.
94/34	R. Bloo F. Kamareddine R. Nederpelt	The Barendregt Cube with Definitions and Generalised Reduction, p. 37.
94/35	J.C.M. Baeten S. Mauw	Delayed choice: an operator for joining Message Sequence Charts, p. 15.
94/36	F. Kamareddine R. Nederpelt	Canonical typing and Π-conversion in the Barendregt Cube, p. 19.
94/37	T. Basten R. Bol M. Voorhoeve	Simulating and Analyzing Railway Interlockings in ExSpect, p. 30.
94/38	A. Bijlsma C.S. Scholten	Point-free substitution, p. 10.

94/39	A. Blokhuis T. Kloks		On the equivalence covering number of splitgraphs, p. 4.
94/40	D. Alstein		Distributed Consensus and Hard Real-Time Systems, p. 34.
94/41	T. Kloks D. Kratsch		Computing a perfect edge without vertex elimination ordering of a chordal bipartite graph, p. 6.
94/42	J. Engelfriet J.J. Vereijken		Concatenation of Graphs, p. 7.
94/43	R.C. Backhouse M. Bijsterveld		Category Theory as Coherently Constructive Lattice Theory: An Illustration, p. 35.
94/44	E. Brinksma R. Gerth W. Janssen S. Katz M. Poel C. Rump	J. Davies S. Graf B. Jonsson G. Lowe A. Pnueli J. Zwiers	Verifying Sequentially Consistent Memory, p. 160
94/45	G.J. Houben		Tutorial voor de ExSpect-bibliotheek voor "Administratieve Logistiek", p. 43.
94/46	R. Bloo F. Kamareddine R. Nederpelt		The λ-cube with classes of terms modulo conversion, p. 16.
94/47	R. Bloo F. Kamareddine R. Nederpelt		On Π-conversion in Type Theory, p. 12.
94/48	Mathematics of Program Construction Group		Fixed-Point Calculus, p. 11.
94/49	J.C.M. Baeten J.A. Bergstra		Process Algebra with Propositional Signals, p. 25.
94/50	H. Geuvers		A short and flexible proof of Strong Normalazation for the Calculus of Constructions, p. 27.
94/51	T. Kloks D. Kratsch H. Müller		Listing simplicial vertices and recognizing diamond-free graphs, p. 4.
94/52	W. Penczek R. Kuiper		Traces and Logic, p. 81
94/53	R. Gerth R. Kuiper D. Peled W. Penczek		A Partial Order Approach to Branching Time Logic Model Checking, p. 20.
95/01	J.J. Lukkien		The Construction of a small CommunicationLibrary, p.16.
95/02	M. Bezem R. Bol J.F. Groote		Formalizing Process Algebraic Verifications in the Calculus of Constructions, p.49.
95/03	J.C.M. Baeten C. Verhoef		Concrete process algebra, p. 134.
95/04	J. Hidders		An Isotopic Invariant for Planar Drawings of Connected Planar Graphs, p. 9.
95/05	P. Severi		A Type Inference Algorithm for Pure Type Systems, p.20.
95/06	T.W.M. Vossen M.G.A. Verhoeven H.M.M. ten Eikelder E.H.L. Aarts		A Quantitative Analysis of Iterated Local Search, p.23.
95/07	G.A.M. de Bruyn O.S. van Roosmalen		Drawing Execution Graphs by Parsing, p. 10.
95/08	R. Bloo		Preservation of Strong Normalisation for Explicit Substitution, p. 12.
95/09	J.C.M. Baeten J.A. Bergstra		Discrete Time Process Algebra, p. 20
95/10	R.C. Backhouse R. Verhoeven O. Weber		MathJpad: A System for On-Line Prepararation of Mathematical Documents, p. 15

95/11	R. Seljée	Deductive Database Systems and integrity constraint checking, p. 36.
95/12	S. Mauw and M. Reniers	Empty Interworkings and Refinement Semantics of Interworkings Revised, p. 19.
95/13	B.W. Watson and G. Zwaan	A taxonomy of sublinear multiple keyword pattern matching algorithms, p. 26.
95/14	A. Ponse, C. Verhoef, S.F.M. Vlijmen (eds.)	De proceedings: ACP'95, p.
95/15	P. Niebert and W. Penczek	On the Connection of Partial Order Logics and Partial Order Reduction Methods, p. 12.
95/16	D. Dams, O. Grumberg, R. Gerth	Abstract Interpretation of Reactive Systems: Preservation of CTL*, p. 27.
95/17	S. Mauw and E.A. van der Meulen	Specification of tools for Message Sequence Charts, p. 36.
95/18	F. Kamareddine and T. Laan	A Reflection on Russell's Ramified Types and Kripke's Hierarchy of Truths, p. 14.
95/19	J.C.M. Baeten and J.A. Bergstra	Discrete Time Process Algebra with Abstraction, p. 15.
95/20	F. van Raamsdonk and P. Severi	On Normalisation, p. 33.
95/21	A. van Deursen	Axiomatizing Early and Late Input by Variable Elimination, p. 44.
95/22	B. Arnold, A. v. Deursen, M. Res	An Algebraic Specification of a Language for Describing Financial Products, p. 11.
95/23	W.M.P. van der Aalst	Petri net based scheduling, p. 20.
95/24	F.P.M. Dignum, W.P.M. Nuijten, L.M.A. Janssen	Solving a Time Tabling Problem by Constraint Satisfaction, p. 14.
95/25	L. Feijs	Synchronous Sequence Charts In Action, p. 36.
95/26	W.M.P. van der Aalst	A Class of Petri nets for modeling and analyzing business processes, p. 24.
95/27	P.D.V. van der Stok, J. van der Wal	Proceedings of the Real-Time Database Workshop, p. 106.
95/28	W. Fokkink, C. Verhoef	A Conservative Look at term Deduction Systems with Variable Binding, p. 29.
95/29	H. Jurjus	On Nesting of a Nonmonotonic Conditional, p. 14
95/30	J. Hidders, C. Hoskens, J. Paredaens	The Formal Model of a Pattern Browsing Technique, p.24.
95/31	P. Kelb, D. Dams and R. Gerth	Practical Symbolic Model Checking of the full μ-calculus using Compositional Abstractions, p. 17.
95/32	W.M.P. van der Aalst	Handboek simulatie, p. 51.
95/33	J. Engelfriet and JJ. Vereijken	Context-Free Graph Grammars and Concatenation of Graphs, p. 35.
95/34	J. Zwanenburg	Record concatenation with intersection types, p. 46.
95/35	T. Basten and M. Voorhoeve	An algebraic semantics for hierarchical P/T Nets, p. 32.
96/01	M. Voorhoeve and T. Basten	Process Algebra with Autonomous Actions, p. 12.
96/02	P. de Bra and A. Aerts	Multi-User Publishing in the Web: DreSS, A Document Repository Service Station, p. 12
96/03	W.M.P. van der Aalst	Parallel Computation of Reachable Dead States in a Free-choice Petri Net, p. 26.
96/04	S. Mauw	Example specifications in phi-SDL.
96/05	T. Basten and W.M.P. v.d. Aalst	A Process-Algebraic Approach to Life-Cycle Inheritance Inheritance = Encapsulation + Abstraction, p. 15.
96/06	W.M.P. van der Aalst and T. Basten	Life-Cycle Inheritance A Petri-Net-Based Approach, p. 18.