State/event net equivalance

Eindhoven University of Technology
Department of Mathematics and Computing Science

State / Event Net Equivalence

by

M. Voorhoeve

98/02

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 Reports 98/02
Eindhoven, May 1998

State/Event Net Equivalence

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

Eindhoven University of Technology
POB 513, 5600MB Eindhoven, the Netherlands

Abstract

The paper is concerned with P/T nets with labeled and unlabeled places and transitions. An equivalence relation is defined for such nets, which abstracts from unlabeled nodes, generalizes place bisimilarity, preserves their branching structure and is a congruence for merge, relabeling, synchronization and refinement. The relation becomes a state-oriented one when all transitions are unlabeled and an event-oriented one when places are unlabeled.

Keywords: Petri nets, Concurrency, Bisimulation, Action Refinement.

1 Introduction

This paper addresses the analysis and verification of concurrent systems. A concurrent system consists of *states* and *events*. The system has an *initial state* and *events* cause it to move from state to state.

The properties of systems are often either event or state based. A typical event based property is *liveness*, the fact that no event will ever become excluded from the system's possible future behavior. Opposed to liveness is *deadlock*, the fact that no event can occur. A typical state based property is *boundedness*, the fact that the set of reachable states of a system is finite.

Many system formalisms are biased towards only one kind of properties; CCS [17] and ACP [3] concentrate on events whereas modal logics [16] and UNITY [6] concentrate on states. It has been shown in [7] that either one-sided approach has the same expressiveness. However, a one-sided approach enforces constructions that are often not very intuitive: states have to be modeled by future events or future events by states.

We therefore believe that two-sided languages and formalisms for concurrency are needed for modeling, validating and verifying concurrent systems. Users should be able to *abstract* from any aspect of the system that is not important to them. The one-sided approaches should reappear by abstracting from either the state or the event aspects.

Petri nets, e.g. P/T (Place/Transition) nets offer a framework for concurrent systems that allows such a two-sided approach. A P/T net is a graph having nodes that are state-related (places) or event-related (transitions). So it exhibits the state-event duality mentioned above. Petri-net models can be constructed directly, as advocated e.g. by Jensen [15]. Nets can also be defined indirectly as models for other formalisms, e.g. PBC (Petri Box Calculus [5]).

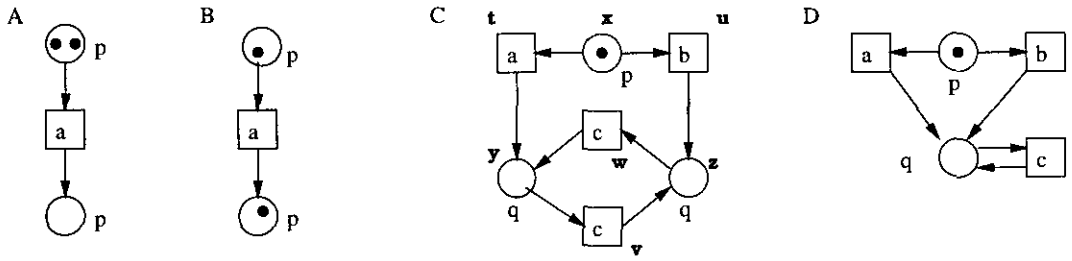

Figure 1: Similar nets

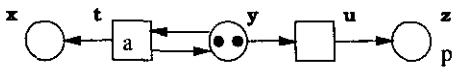

Figure 2: A net with unlabeled nodes

Nets that model concurrent systems have *labeled* nodes. These labels link the places and transitions in the net to states and events in the modeled system. A token in a place x with label p in the net corresponds to a p object in the system and the firing of a transition t with label a in the net corresponds to an a action in the system. This is extended to states (several tokens) and events (concurrent actions).

Different nodes may possess the same label. The event or state components corresponding to these nodes may be indistinguishable. The nets A and B in Figure 1 have states that look equivalent (two p objects). Their future behavior is however not the same: A allows two a actions whereas B allows only one. So the systems modeled by these nets are different.

Nets C and D in this Figure 1 model one and the same concurrent system. (In the figure, nodes have both labels and identifiers; the identifiers appear in a bold font.) The initial state of both nets has label p and allows for two different events (labeled a and b) leading to a q-labeled state allowing for an infinite succession of c-labeled events. The fact that C has two q-labeled places (**y** and **z**) and two c-labeled transitions (**v** and **w**) does not matter. These places and transitions cannot be distinguished by observing the visible state and future behavior.

One can *abstract* from certain nodes in the net by leaving them unlabeled. [1] Unlabeled places are excluded from the visible state of the modeled system, although tokens in these places can manifest themselves by directly or indirectly allowing visible events. Likewise, the firing of unlabeled transitions is not a visible event, but its occurrence may become manifest if it consumes or produces manifest tokens.

In Figure 2 we have labeled (**z**,**t**) and unlabeled (**x**,**y**,**u**) nodes. Tokens in **y** are manifest since they allow a visible event (with label a). The event **u** is manifest since it produces a visible token (with label p). Tokens in **x** are not manifest as they do not allow any event.

In event-oriented approaches (ACP, CCS), states are abstracted from. Thus, nets that model ACP or CCS terms have unlabeled places. In PBC all places except "entry" and

[1] In many formalisms for concurrent systems the "silent" label τ is used instead.

"exit" places are unlabeled. UNITY abstracts from events; thus corresponding nets have unlabeled transitions.

When verifying properties of a concurrent system (modeled as a net), one usually abstracts from many actions and objects. After abstraction, the net may be reduced to an equivalent smaller net, for which the desired properties become easier to verify. Of course, the equivalence relation should conserve the properties that are being verified.

Modeling nets in a compositional way requires the use of operators that combine and modify nets. Nets that are equivalent should stay so when the same operator is applied to both. The equivalence is then said to be a *congruence* for such an operator. Congruence properties allow verification by composition, which is often the only viable way for large nets.

In this paper we define state-event (SE) bisimilarity, an equivalence relation for labeled P/T nets that relates nets like C and D in Figure 1. We are aware of the fact that there exist many such relations with their respective merits and demerits (c.f. [9], [18]). However, SE bisimilarity substantially differs from other equivalences. We present the requirements for a true state-event equivalence relation.

i) It should respect both state and event labels, while allowing to abstract from any number of places and/or transitions. By abstracting from all places, it should become an event-oriented equivalence and by abstracting from transitions it should become a state-oriented one.

ii) It should preserve the *branching structure* of the system, i.e. the options for continuation in each state.

iii) It should preserve its *concurrency structure*. If two transitions are in conflict in a net, and hence cannot fire concurrently, a similar conflict should be present in equivalent nets.

iv) It should be a congruence for operators like merge and synchronization.

v) It should allow reductions of a net modulo the equivalence relation by simple rules.

To our knowledge, the equivalence relations partially satisfying the first requirement are place bisimilarity [2] and (forward) interface equivalence [21]. Weak place bisimilarity [1] allows abstraction from transitions, whereas interface equivalence allows abstraction from places.

Event-oriented equivalences that satisfy the second requirement are the various bisimilarities. ST bisimilarity [12], [19] also satisfies the third and fourth requirement. Place bisimilarity has a very appealing reduction algorithm. For bounded nets, "interleaving" bisimilarities share this property, but they require the costly construction of the whole state space of the net.

SE bisimilarity does not allow preservation of the *causal structure* of nets, as shown in Figure 3. In this figure, net A has two unlabeled nodes and two labeled ones. When the

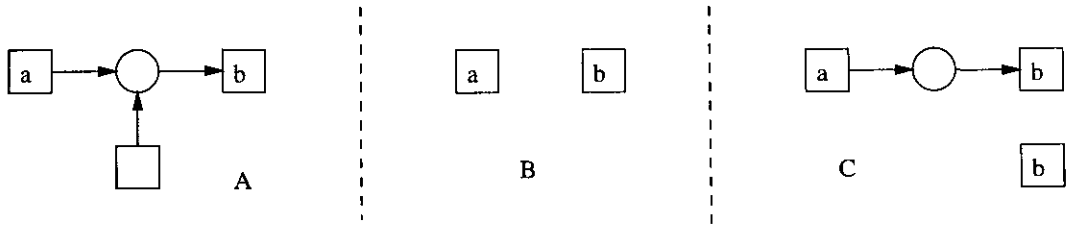

Figure 3: SE bisimilar nets with different causal structure

b transition fires, this has been caused either by the a transition by or the unlabeled one. The same causal structure is present in C, but not in B. All three nets are SE equivalent. Causality respecting notions, like history preserving bisimilarity [11] thus are stronger than SE bisimilarity in this respect. However, it is possible to retain causality by *not* abstracting from causes, i.e. by labeling the places that embody the causal relations that one wants to preserve. By abstracting from a cause, one must accept that causal relations may get lost modulo SE bisimilarity.

In the remainder of the paper we start with a motivating example, indicating the kind of reductions that SE bisimilarity should allow. A short section introduces some general notations about relations and bags. We then define our equivalence relation and establish its properties. We conclude by discussing some possibilities of SE bisimilarity for modeling and verification of concurrent systems.

2 Example

In this section we present a toy example inspired by automated manufacturing. In the left-hand net in Figure 4, the processing and serving of food at a burger restaurant is depicted. Customers can order a meal (transition `order`), which is noted on two pieces of paper. One note (place `ao` for *administrative order*) is transferred to the cashier, where the meal can be paid (transition `pay`). The copy (place `po` for *production order*) is transferred to the kitchen. In the kitchen, upon receiving an order, raw meat (`rm`) is fried and the fried meat (`fm`) is assembled with two slices of tomato (`ts`) and the two halves of a bun (`ub,lb`). The output is a burger meal (place `burger`). Raw meat, tomatoes and buns are replenished (transitions `repm,rept,repb`); tomatoes are sliced in eight and buns are cut.

The customer protocol of the net is obtained by abstracting from all non-customer nodes, retaining only the labels `order,pay,burger`. The net thus relabeled can be reduced modulo SE bisimilarity to the net in the upper right quarter. Note that paying and the burger becoming available are *concurrent* events, so that e.g. a parent can pay while a child is being served.

The lower right quarter is the reduction w.r.t. the logistics protocol. The reduction clearly shows how the number of burgers manufactured depends on the number of orders and the amount of meat, tomatoes and bread replenished. In subsection 4.1 we establish directly

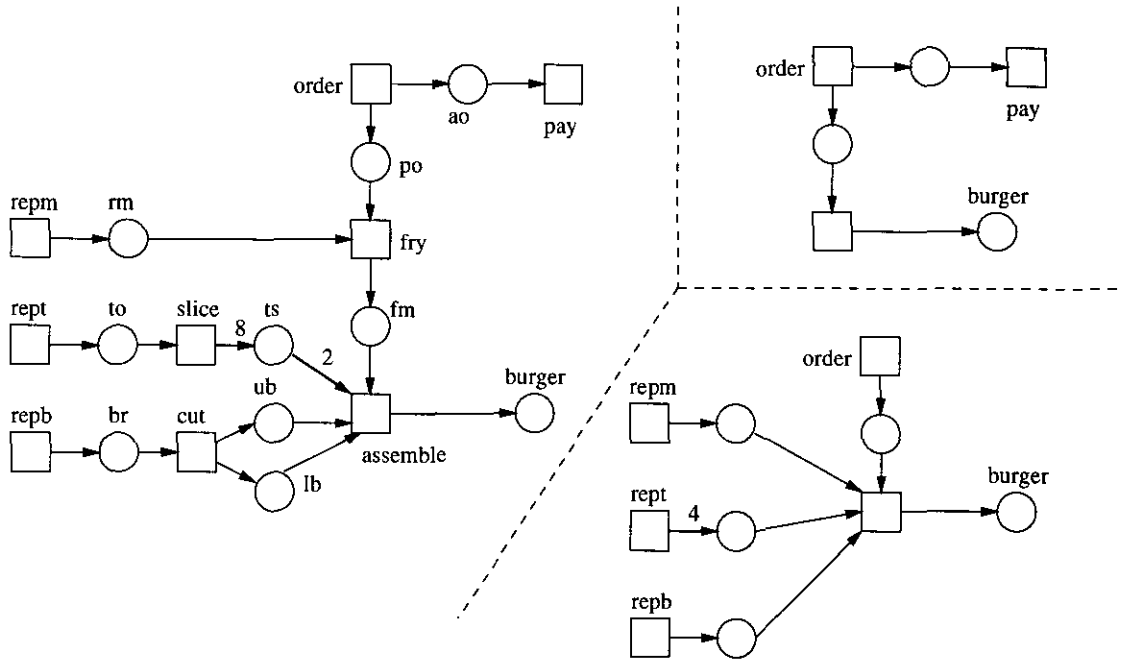

Figure 4: Example Net: Burger restaurant

the equivalences of the nets and in subsection 4.5 we do so indirectly by applying reduction rules.

One could re-engineer the process, for instance by allowing to fry a limited amount of meat in advance, i.e. without order. Re-engineering should not affect the customer and logistics protocol, which can be verified by reducing the re-engineered processes after abstraction modulo SE bisimilarity and obtaining the same reduced nets.

3 Bags and relations

We assume the usual notation about sets and functions, which are considered a subclass of the binary relations defined below. Unless mentioned otherwise, $A, B \ldots$ are sets and $a, b \ldots$ elements of these sets. We use the minus sign for set difference. We can write $A - a$ instead of $A - \{a\}$ if there is no confusion possible.

A *relation* between A and B is a subset of $A \times B$. These relations are partially ordered by means of set inclusion. An element of a relation is denoted with brackets, e.g. (a, b). Table 1 shows some operators for relations. We have adopted the semicolon notation for relation composition to avoid problems with function composition. In the table, n is a positive natural number, R, S are arbitrary relations and A an arbitrary set.

We write $a\ R\ b$ instead of $(a, b) \in R$. A relation R is a *function* iff $\forall (a, b), (c, d) \in R : a = c \Rightarrow b = d$. Functions are defined as sets of pairs. For a function f, $(a, b) \in f$, $a\ f\ b$ and $f(a) = b$ mean the same thing. Note that for functions f, g, $(f; g)(x) = g(f(x))$,

Expression	Meaning	Remarks
$R; S$	$\{(a,b) \mid \exists x : (a,x) \in R \wedge (x,b) \in S\}$	Composition
R^n	$R; \ldots ; R$	Relation to n-th power
R^+	$\bigcup\{R^i \mid i > 0\}$	Transitive closure
R^{-1}	$\{(a,b) \mid (b,a) \in R\}$	Inverse
$\mathcal{D}_A$	$\{(a,a) \mid a \in A\}$	Diagonal
$\mathrm{dom}(R)$	$\{a \mid \exists x : (a,x) \in R\}$	Domain
$\mathrm{ran}(R)$	$\{a \mid \exists x : (x,a) \in R\}$	Range (Codomain)

Table 1: Operators for relations

which is an awkward consequence of tradition. $A \to B$ denotes the set of total functions f with $\mathrm{dom}(f) = A$ and $\mathrm{ran}(f) \subseteq B$ and $A \hookrightarrow B$ the set of partial functions. A function $f \in A \to B$ such that $f^{-1} \in B \to A$ is called an *injection.*

Sequences are noted between $\langle\rangle$ signs. A *D-sequence* (D stands for *difference*) is a sequence of which any consecutive elements are different. The operator $\bullet$ combines two D-sequences to a new D-sequence: $\langle a_1, \ldots, a_n\rangle \bullet \langle b_1, \ldots, b_m\rangle$ equals $\langle a_1, \ldots, a_n, b_1, \ldots b_m\rangle$ if $a_n \neq b_1$ and $\langle a_1, \ldots, a_{n-1}, b_1, \ldots b_m\rangle$ otherwise. (Note that $a_{n-1} \neq a_n = b_1$ in this case.) The set $\mathbf{D}(A)$ is the set of D-sequences with elements from A.

We will use sequences to trace states of a dynamic system. The sequences $\langle a, a\rangle$ and $\langle a\rangle$ are equivalent. Instead of D-sequences and the $\bullet$ operator we could also have used equivalence classes of ordinary sequences and sequence composition, for which the equivalence relation is a congruence.

Bags are functions with range $\mathbb{N}$. For the sake of convenience, we assume the existence of a universe $\mathcal{U}$ that contains all the domains of bags to be treated here. Bags then can be considered to be elements of $\mathcal{U} \to \mathbb{N}$. The set $\mathcal{B}(C)$ of bags with carrier $C \subseteq \mathcal{U}$ is defined as $\{a \in \mathcal{U} \to \mathbb{N} : \forall x \in (\mathcal{U} - C) : a(x) = 0\}$. If a bag A has a finite carrier C, it has a finite size $|A|$ defined as $\Sigma((a,n) \in A \wedge a \in C : n)$.

Operators for bags are defined in Table 2, where $A, B \in \mathcal{B}(\mathcal{U}); a \in \mathcal{U}; n \in \mathbb{N}$; I is an index set; $A_i \in \mathcal{B}(\mathcal{U})$ for $i \in I$. The $\in$ bag element operator overrides the standard set element operator (a bag is a set of pairs).

As usual, we denote the inverse of $\leq$ by $\geq$. We sometimes write $\Sigma_Q x$ instead of $\Sigma(Q : x)$. The priority rules for addition, division and multiplication are assumed. We may omit the $+$ sign when writing down bag examples. Also we may write a instead of a^1 when the ambiguity between element and singleton bag is clear from the context. So the bag $a^2 + b^1 + c^3$ may be represented as a^2bc^3.

We conclude this section with a theorem on bags and relations which is a consequence of P. Hall's marriage theorem ([14], page 48). We first state Hall's theorem. It gives a necessary and sufficient condition for a set of sets $\{S_i \mid i \in I\}$ to have a distinct element $\psi(i)$ in each member set S_i.

Theorem 1 *Let I be a finite index set and S_i be a set for each $i \in I$. Then there exists*

Expression	Condition	Meaning	Remarks
$\mathbf{0}$		$\{(x,0) \mid x \in \mathcal{U}\}$	empty bag
a^n		$\{(x,0) \mid x \in (\mathcal{U}-a)\} \cup \{(a,n)\}$	one-point bag
$a \in A$		$A(a) > 0$	bag element
$A+B$		$\{(x, A(x)+B(x)) \mid x \in \mathcal{U}\}$	bag sum
$\Sigma(i \in I : A_i)$		$\{(x, \Sigma(i \in I : A_i(x))) \mid x \in \mathcal{U}\}$	bag sum
$A-B$		$\{(x, \max(0, A(x)-B(x))) \mid x \in \mathcal{U}\}$	bag difference
$A \cap B$		$\{(x, \min(A(x), B(x))) \mid x \in \mathcal{U}\}$	bag intersection
$A \cup B$		$\{(x, \max(A(x), B(x))) \mid x \in \mathcal{U}\}$	bag union
$A \leq B$		$\forall x \in \mathcal{U} : A(x) \leq B(x)$	bag inclusion

Table 2: Bag operators

an injection ψ with domain I such that $\psi(i) \in S_i$ iff $\forall J \subseteq I : |\bigcup\{S_i \mid i \in J\}| \geq |J|$.

The necessity is trivial, since the restriction of ψ to J must have domain and range of the same size. We apply this theorem to bags, giving some auxiliary notions first.

Definition 1 *Let $R \subseteq \mathcal{U} \times \mathcal{U}$ be a relation. The relation $\bar{R}$ between bags is defined as follows. $A\ \bar{R}\ B$ iff A, B can be written in the form $\Sigma_{i\in I} a_i^1, \Sigma_{i\in I} b_i^1$ respectively, where I is an index set, such that $\forall i \in I : a_i\ R\ b_i$.*

The $\bar{R}$ relation is the "natural" lifting of R to bags. Two bags are $\bar{R}$ related iff each one can be split up into singletons that are R-related. For example, if $R = \{(a,b),(a,c),(d,c)\}$, then $a^2d\ \bar{R}\ bc^2$, since we can take $I = \{1,2,3\}, a_1 = a_2 = a, a_3 = d, b_1 = b, b_2 = b_3 = c$. On the other hand, $(ad^2, b^2c) \notin \bar{R}$. We will apply Hall's theorem by giving a condition that allows one to deduce that bags are $\bar{R}$-related.

Corollary 1 *Let $R \subseteq \mathcal{U} \times \mathcal{U}$ be a finite relation. Let ρ be a relation between bags satisfying $A\ \rho\ B \Rightarrow (\forall a \in A : \exists b \in B : a\ R\ b) \wedge \forall b \in B : \exists c \in A : (A-c)\ \rho\ (B-b)$. Then $\rho \subseteq \bar{R}$.*

Proof: Let A, B be bags such that $A\ \rho\ B$. We shall prove that $A\ \bar{R}\ B$.
Since R is finite, the carrier of A is finite. We can thus write A in the form $\Sigma_{k\in K} a_k^1$, for some index set K. Clearly, $|K| = |A|$, so K is finite. Write B as $\Sigma_{i\in I} b_i^1$. Using induction on $|K|$ and the conditions on ρ we have $|K| = |I|$. For $i \in I$ we set $S_i = \{k \in K \mid a_k\ R\ b_i\}$. Let $J \subseteq I$. Let $B' = \Sigma_{i\in J} b_i^1$. By the second condition on ρ, there exists an $A' \leq A$ such that $A'\ \rho\ B'$. So there exists an $L \subseteq K$ such that $A' = \Sigma_{k\in L} a_k^1$. Again, $|J| = |L|$. On the other hand, by the first condition on ρ, $L \subseteq \bigcup\{S_i \mid i \in J\}$, so the condition for Hall's theorem holds.
Hence, there exists an injection ψ as in Hall's theorem. As $\psi(i) \in S_i$ we have that $\psi(i)$ is equal to some k with $a_k\ R\ b_i$. Set $\alpha_i = a_{\psi(i)}$. We have $A = \Sigma_{i\in I} \alpha_i^1$ so $A\ \bar{R}\ B$. □

4 Labeled P/T nets

We presuppose a set $\mathcal{L}$ of *labels* and an infinite set $\mathcal{N}$ of nodes. A labeled place-transition net (LPT net) is a bipartite directed graph (multiple edges allowed) with labeled and unlabeled nodes. For notational simplicity, we introduce the "silent" label $\tau \notin \mathcal{L}$ to make the labeling total. We abbreviate $\mathcal{L} \cup \{\tau\}$ by $\mathcal{L}_\tau$.

Definition 2 *An LPT net L is a five-tuple (P, T, I, O, ℓ), where $P, T \subseteq \mathcal{N}$ are finite disjoint sets; $I, O \in T \to \mathcal{B}(P)$ and $\ell \in (T \cup P) \to \mathcal{L}_\tau$, representing the respective places, transitions, input function, output function and labeling function.*
A process *is a pair (L, E) where $L = (P, T, I, O, \ell)$ is an LPT net and $E \in \mathcal{B}(P)$ its* initial *(entry) state.*

In the remainder of the paper, "net" will mean "LPT net". The components of a net are denoted by subscripting them, so e.g. P_L is the set of places of net L. If the net is clear from the context, these subscripts may be omitted. The set $P \cup T$ can be abbreviated by N (nodes). The nodes n with $\ell(n) \neq \tau$ are called the labeled nodes, as τ is not a true label.

There exists a relation called *isomorphism* between LPT nets. An isomorphism is a bijection between the nodes of two nets that preserves labeling (ℓ) and structure (I, O).

Nets are represented by directed graphs like in the preceding figures. Transitions are represented by squares and places by circles. If e.g. $I(t)(p) = n > 0$ for a certain transition t and place p, then an arc is drawn with weight n from p to t. For O, the arcs are inverted. Weights one are omitted for clarity.

4.1 SE bisimilarity

Let L be a net. For convenience, we omit subscripts L. The concept of SE bisimilarity stems from the notion about the dynamic behavior of nets, where actions have a duration, as in [12]. Each action can be started and then committed. Between these events, the action is executed. The fact that actions are not atomic allows one to model concurrency without causality: two actions are concurrent in a given state if one can be started immediately after starting the other, so that both are executed simultaneously. We include the possibility that actions that are being executed can be revoked. Actions cannot be revoked once they are committed.

A *state* of the net is an element of $\mathcal{B}(N)$, which can be written as a sum $x + y$ where $x \in \mathcal{B}(P)$ and $y \in \mathcal{B}(T)$. The x component contains the passive objects in the state and y represents the actions that are concurrently being executed. An action consumes certain objects and produces new objects, as specified by the I and O functions. The state can be altered by starting an action, which is the s relation defined below. The consumed objects are removed from the state and the action is added. The relation s^{-1} embodies the rollback relation. The commit relation c removes an action from the state and adds the produced objects.

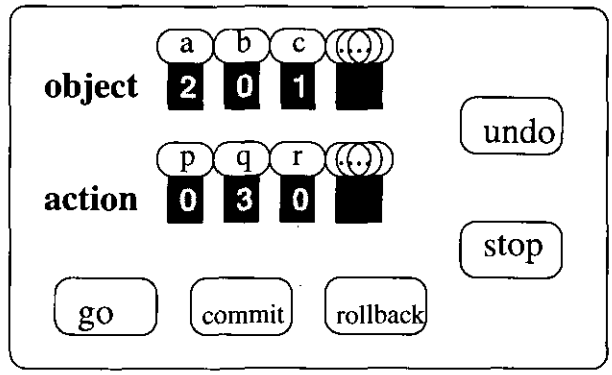

Figure 5: Testing LPT nets

The $\gg$ relation removes a *visible* object or action from the state. This relation plays a vital part in the definition of SE bisimilarity.

In Figure 5, we show a hypothetical LPT net testing device that illustrates some concepts sketched above. The device has two rows ("object" and "action") of displayed numbers, with labeled buttons on top of each display. These buttons are supposed to contain all possible place labels for the "object" row and all possible transition labels for the "action" row. It also contains five larger buttons ("go", "stop", "commit", "rollback" and "undo").

A user of the testing device can insert a process into it. Its state will then be displayed in the "object" row: the number below an a-labeled button will equal the total number of initial tokens in the places with label a.

After inserting a process, a session can start. A session alternates between dynamic and static modes. Initially, the device is in a static mode. By pressing "go", "commit" or "rollback", the mode becomes dynamic and state changes are possible. Each new state reached is displayed in both the "object" and "actions" row: the object labels and the labels of the pending (started bu not yet committed) actions. So in the state displayed in Figure 5, two a-labeled and one c-labeled objects are present and three b-labeled transitions are pending. By pressing "stop", the state (both objects and actions) is frozen and a static mode is reached.

The difference between "go", "commit" and "rollback" is that "go" allows both the starting and committing of transitions, "commit" only allows the committing of pending transitions and "rollback" allows the revoking of pending transitions.

The "undo" and small labeled buttons can be pressed in static mode and leave the system in static mode. A stack of states in static mode is kept, with the initial state at the bottom and the last reached state on top. Pressing "go", "commit" or "rollback" and then "stop" will push a new state onto the stack. Pressing "undo" will pop the top state off the stack. Pressing a labeled button above a display is possible when the displayed number is nonzero. This will cause the removal of one object with the corresponding label. This new state is pushed onto the stack.

All the above reactions to pressing buttons are nondeterministic: any possible reaction

that meets the specifications may occur. The device has even a "fairness warranty": by pressing "undo" and repeating the same sequence of actions sufficiently often, *all* possible reactions that meet the specification will occur.

Definition 3 *We define the following relations between bags of nodes.*

$$\begin{aligned} s &= \{(X + I(t), X + t) \mid t \in T \wedge X \in \mathcal{B}(N)\} \\ c &= \{(X + t, X + O(t)) \mid t \in T \wedge X \in \mathcal{B}(N)\} \\ \gg &= \{(X + n, X) \mid n \in N \wedge \ell(n) \neq \tau \wedge X \in \mathcal{B}(N)\} \end{aligned}$$

The relations γ, ρ between bags of nodes are the reflexive-transitive closures of c, s^{-1} respectively. (The reflexive-transitive closure of a relation R between bags of nodes is $\mathcal{D}_{\mathcal{B}(N)} \cup R^+$).
We derive from the function $\ell \in N \to \mathcal{L}$ the function $\tilde{\ell}$ in $\mathcal{B}(N) \to (\mathcal{B}(\mathcal{L}) \times \mathcal{B}(\mathcal{L}))$ as follows. $\tilde{\ell}(\Sigma_{x \in P} x^{n_x} + \Sigma_{x \in T} x^{n_x}) = (\Sigma(x \in P \wedge \ell(x) \neq \tau : \ell(x)^{n_x}), \Sigma(x \in T \wedge \ell(x) \neq \tau : \ell(x)^{n_x}))$.
For $\sigma \in \mathbf{D}(\mathcal{B}(\mathcal{L}))$, we define the relation $\rhd^\sigma$ as the smallest relation satisfying $A \rhd^{\langle \tilde{\ell}(A) \rangle} A$ and $A \rhd^{\rho \bullet \sigma} B$ iff there exist C, D with $C\ (s \cup c)\ D$ and $A \rhd^\rho C$ and $D \rhd^\sigma B$. We define the reachability relation $\rhd$ as $\bigcup_\sigma \rhd^\sigma$, so $X \rhd X'$ iff there exists a σ such that $X \rhd^\sigma X'$.

We illustrate the notions in the definition that we did not introduce earlier by means of the device in Figure 5. By pressing "go", observing the display and then hitting "stop", the user can observe the $\rhd^\sigma$ relations. The σ corresponds to the sequence of observed states between the first and the last state reached. Likewise, pressing "commit" and "rollback" reflects the respective γ and ρ relations. Hitting the display buttons reveals the $\gg$ relation. If the displayed label belongs to a transition, this relation corresponds to *aborting*: terminating without releasing its input or output objects.
As an example, taking C in Figure 1 and using the identifiers to describe elements of $\mathcal{B}(N_C)$, we have $\mathbf{x^1 t^1}\ c\ \mathbf{x^1 y^1}$, $\mathbf{x^1 t^1}\ s\ \mathbf{t^2}$, $\mathbf{x^1 t^1} \gg \mathbf{x^1}$ and $\mathbf{x^2} \rhd^{\langle (p^2,0) \rangle} \mathbf{x^2}\ s\ \mathbf{xt} \rhd^{\langle (p,a) \rangle} \mathbf{xt}$, so $\mathbf{x^2} \rhd^{\langle (p^2,0),(p,a) \rangle} \mathbf{xt}$. Continuing like this we obtain $\mathbf{x^2} \rhd^{\langle (p^2,0),(p,a),(0,a^2) \rangle} \mathbf{t^2}$. In Figure 2, we have $\mathbf{y^2} \rhd^{\langle (0,0),(0,a),(p,a),(p,0) \rangle} \mathbf{xz}$ and also $\mathbf{y^2} \rhd^{\langle (0,0),(0,a),(0,0),(p,0) \rangle} \mathbf{xz}$.
A *place-only* state is a subset of $\mathcal{B}(P)$. We have $\mathcal{B}(N) - \mathrm{dom}(s^{-1}) = \mathcal{B}(P)$, so the place-only states are those states that cannot be revoked. Every state can reach a place-only state by either committing or revoking its pending transitions. We give a lemma that reflects the "diamond property" of nets. Any state reachable by starting a number of actions and then committing all of them can also be reached by starting and then committing one action at the time.

Lemma 1 *The following inclusions hold: $s; s; c \subseteq s; c; s$ and for $n > 0$, $s^n; c^n \subseteq (s; c)^n$.*

Proof: We first prove that $s; s; c \subseteq s; c; s$. Let $A\ s; s; c\ B$, so there are $C \in \mathcal{B}(N)$ and $t, u, v \in T$ such that $I(t) + I(u) \leq A$, $C = A - I(t) - I(u) + t + u$, $v^1 \leq C$ and $B = C + O(v) - v$. We may assume that $v^1 \leq C - u$ (otherwise, swap t and u). Then $A\ s\ (A - I(t) + t)\ c\ (A - I(t) + O(v) + t - v) = (C + I(u) + O(v) - u - v)\ s\ B$.

Next, we define $r^0 = \mathcal{D}_{\mathcal{B}(N)}$ for any relation r and prove by induction that $s;(s;c)^n \subseteq (s;c)^n;s$ for $n \geq 0$. The case $n = 0$ is clear, whereas for $n > 0$, $s;(s;c)^n = s;(s;c)^{n-1};s;c \subseteq (s;c)^{n-1};s;s;c \subseteq (s;c)^{n-1};s;c;s = (s;c)^n;s$.
Again with induction, we prove the lemma: $s^n;c^n = s;s^{n-1};c^{n-1};c \subseteq s;(s;c)^{n-1};c \subseteq (s;c)^{n-1};s;c = (s;c)^n$. □

We now define state/event (SE) bisimilarity. Suppose that a user obtains a process X modeled as a net and another process Y inside the device in Figure 5. By experimenting with the device, he can try to detect whether Y can behave in a way that X cannot behave, e.g. by starting a transition with a label that does not occur in X. By undoing and repeating the same experiment sufficiently often, the user can detect whether X can behave in a way that Y cannot behave. If the user cannot detect any differences between X and Y, the two processes are SE bisimilar. We give the formal definition below.
Note that $(R^{-1};A) \subseteq (B;R^{-1})$ amounts to stating that to every x, y, x' such that $x\ R\ y$ and $x\ A\ x'$ (so $y\ (R^{-1};A)\ x'$, thus $y\ (B;R^{-1})\ x'$), there must exist a y' such that $y\ B\ y'$ and $x'\ R\ y'$. As the definition involves two nets, the relations $\rhd^\sigma$, $\gg$ and so on are subscripted with the proper net.

Definition 4 *An* SE simulation *from net L to M is a relation $R \in \mathcal{P}(\mathcal{B}(N_L) \times \mathcal{B}(N_M))$ such that for all $\sigma \in \mathbf{D}(\mathcal{B}(\mathcal{L}))$,*

$$\begin{array}{llllll} (trace) & R^{-1};\rhd_L^\sigma \subseteq \rhd_M^\sigma;R^{-1}, & (abort) & R^{-1};\gg_L \subseteq \gg_M;R^{-1}, \\ (commit) & R^{-1};\gamma_L \subseteq \gamma_M;R^{-1}, & (rollback) & R^{-1};\rho_L \subseteq \rho_M;R^{-1}. \end{array}$$

An SE bisimulation *between L, M is an SE simulation from L to M, the inverse of which is an SE simulation from M to L. The nets L and M are* structurally SE bisimilar *(notation $L \sim M$) iff there exists a total surjective SE bisimulation between L and M. The processes (L, E) and (M, F) are* SE bisimilar *(notation $(L, E) \sim (M, F)$) iff there exists an SE bisimulation R such that $E\ R\ F$.*

Observe that if X and Y are related by an SE bisimulation, $\tilde{\ell}(X) = \tilde{\ell}(Y)$ (since $X \rhd^{\tilde{\ell}(X)} X$). The relation R between $\mathcal{B}(N_C)$ and $\mathcal{B}(N_D)$ in Figure 1 defined by $X\ R\ Y \Leftrightarrow \tilde{\ell}_C(X) = \tilde{\ell}_D(Y)$ is a total and surjective SE bisimulation, so these nets are structurally SE bisimilar. Likewise for the nets in Figure 3.
We give bisimulations for our running example in Figure 4. We use conditions (predicates w.r.t. pairs of states) to describe the bisimulations; states are related iff they satisfy all the conditions. The conditions all have the form that linear combination of weights of the nodes of one net must be equal to another linear combination of weights of the nodes in the other net.
In the diagrams the conditions are depicted by dashed lines: if short dashed lines are drawn between nodes, say, a, b, c of one net to a common point and this point is connected to a node d of the other net, this signifies that the weight of d must be equal to the sum of the weights of a, b and c. The short lines may have multipliers attached to them. For instance

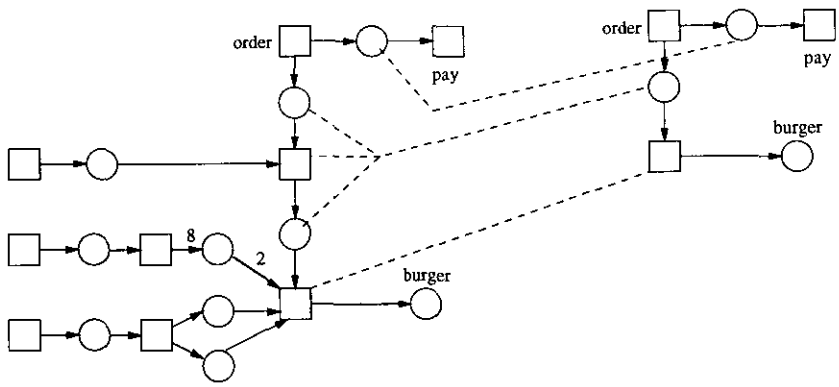

Figure 6: Burger example: client protocol

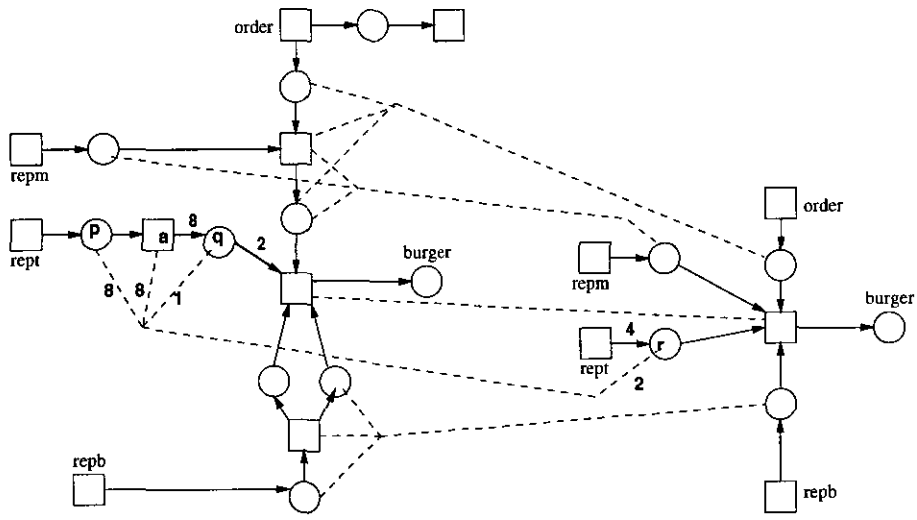

Figure 7: Burger example: logistics protocol

Figure 7 contains a condition saying that twice the weight of **c** equals the sum of eight times the weight of **a**, eight times the weight of **p** and the weight of **b**. In the figures, the conditions that equally labeled nodes have the same weight are omitted.

After relabeling, the relation between states of the left-hand and upper right-hand processes of Figure 4, given by the conditions in Figure 6 is an SE bisimulation. Figure 7 defines an SE bisimulation for the logistics protocol in the same way. The initial states are empty in both processes of both figures.

Process A is not bisimilar to B, because there exists no relation satisfying the trace condition. From the initial (empty) state, states can be reached with label p^1 in both nets. In A, in such a state both b and c labeled transitions can start, but in B only one of them can start.

In Figure 8 process pairs are given that are *not* SE bisimilar. The processes C and D are not SE bisimilar because the rollback condition cannot be satisfied. In D, a state is

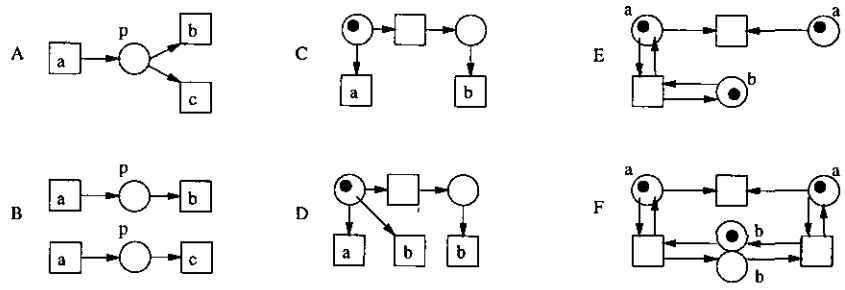

Figure 8: Non SE bisimilar process pairs

reachable where a b-labeled transition is firing and by undoing an a event becomes possible. A similar state does not exist in C.

No relations between processes E and F comply with the abort condition. In E, an a-labeled object can be removed resulting in an $ab/\mathbf{0}$ cycle. This is not possible in F.

We have not found examples like the ones above for the commit condition. It seems that the existence of a relation between net states satisfying the three other conditions implies the existence of an SE bisimulation with the same domain and range.

Theorem 2 *SE bisimilarity of processes and structural SE bisimilarity of nets are equivalence relations.*

Proof: The identity relation $D_{\mathcal{B}}(N)$ is an SE bisimulation for any given net, proving reflexivity. If Q is an SE bisimulation, then Q^{-1} is an SE bisimulation too, proving symmetry. Let K, L and M be nets. Let Q, R be SE bisimulations between K and L, L and M respectively. Then $Q;R$ is an SE bisimulation between K and M. The simulation properties are straightforward by formula manipulation, e.g.:
$(Q;R)^{-1};\rhd^{\sigma}_{K} = R^{-1};(Q^{-1};\rhd^{\sigma}_{K}) \subseteq R^{-1};\rhd^{\sigma}_{L};Q^{-1} \subseteq (\rhd^{\sigma}_{M};R^{-1});Q^{-1} = \rhd^{\sigma}_{M};(Q;R)^{-1}$. So SE bisimilarity is transitive. □

A net isomorphism defines a total and surjective SE bisimulation, so isomorphic nets are structurally SE bisimilar.

4.2 Connection with place bisimilarity

In this part, we prove that SE bisimilarity generalizes the idea of place bisimilarity [2], in which individual places have to be related to one another. In [20], this idea has been extended to transitions as well. We shall use the term "node bisimilarity" for this equivalence relation. We give a series of definitions to show how these ideas are related.

Definition 5 *Let L be an LPT net. For any $\alpha \in \mathcal{L}_{\tau}$, the relation $\xrightarrow{\alpha} \subseteq \mathcal{B}(P_L) \times \mathcal{B}(P_L)$ is defined by $X \xrightarrow{\alpha} Y$ iff there exists a $t \in T_L$ with $\ell(t) = \alpha$ such that $I(t) \leq X$ and $Y = X - I(t) + O(t)$. (So X s $X - I(t) + t$ c Y).*

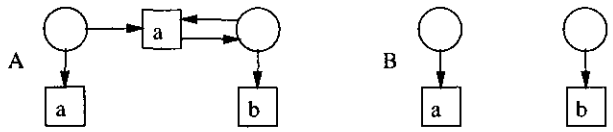

Figure 9: Place bisimilar nets

Definition 6 *Let L and M be LPT nets. A node bisimulation between L and M is a relation $r \subseteq N_L \times N_M$ such that $\bar{r}$ is an SE bisimulation.*
A strong bisimulation between L and M is a relation $R \subseteq \mathcal{B}(P_L) \times \mathcal{B}(P_M)$ such that for any $a \in L_\tau$, $R^{-1}; \xrightarrow{a}_L \subseteq \xrightarrow{a}_M; R^{-1}$ and $R; \xrightarrow{a}_M \subseteq \xrightarrow{a}_L; R$.
A place bisimulation between L and M is a relation $r \subseteq P_L \times P_M$ such that $\bar{r}$ is a strong bisimulation.
L and M are structurally node/strongly/place bisimilar iff there exists a total and surjective node/strong/place bisimulation between them. Processes (L, E) and (M, F) are node/strongly/place bisimilar iff there exists a node/strong/place bisimulation R between L and M such that $E\ R\ F$.

Lemma 2 *Let L, M be LPT nets without τ-labeled transitions. Then a node bisimulation between L and M is a place bisimulation between them.*

Proof: Let r be the node bisimulation and let $R = \bar{r}$. We shall prove that $R^{-1}; \xrightarrow{a}_L \subseteq \xrightarrow{a}_M; R^{-1}$. The other condition follows by symmetry. Suppose $X, X' \in \mathcal{B}(P_L); Y \in \mathcal{B}(P_M)$ such that $X\ R\ Y$ and $X \xrightarrow{a} X'$. Then there exists a $X'' \in \mathcal{B}(N_L)$ such that $X \rhd X'' \rhd X'$ and $\tilde{\ell}(X'')$, $\tilde{\ell}(X)$ and $\tilde{\ell}(X')$ have the form $(\xi, \mathbf{0})$, (η, a) and $(\zeta, \mathbf{0})$ respectively. Since R is an SE bisimulation, we can find Y', Y'' such that $X'\ R\ Y'$, $X''\ R\ Y''$, $Y \rhd Y'' \rhd Y''$ and $\tilde{\ell}(Y'')$, $\tilde{\ell}(Y)$ and $\tilde{\ell}(Y')$ have the form $(\xi, \mathbf{0})$, (η, a) and $(\zeta, \mathbf{0})$ respectively. This yields $Y \xrightarrow{a} Y'$ since M has no τ-labeled transitions. □

Figure 9 shows nets that are structurally place bisimilar, but not node bisimilar. In net A a conflict is possible between a and b actions, which has disappeared in net B. Place bisimilarity does not preserve the concurrency structure.

We shall prove that an SE bisimulation R defines a relation r between the *labeled* nodes of the nets involved such that $R = \bar{r}$ when restricted to labeled nodes only. So *totally* labeled nets (no τ labels) are node bisimilar, and thus place bisimilar.

In [1], another generalization ("weak" place bisimilarity) is given that also deals with the special nature of unlabeled (or τ-labeled) nodes. The main difference with SE bisimilarity is that in weak place bisimilarity individual places still have to be related to one another.

Let L be a net with nodes N and labeled nodes N'. We define the projection $\pi : \mathcal{B}(N) \to \mathcal{B}(N')$ by $\pi(X) = \Sigma_{x \in N'} x^{X(x)}$. We recollect the definition of the bar operator in Definition 1.

Lemma 3 *Let L, M be nets and R an SE bisimulation between them. Then there exists a relation r between the labeled nodes N'_L, N'_M of L and M respectively such that $A\ R\ B \Rightarrow \pi(A)\ \bar{r}\ \pi(B)$.*

Proof: Let L, M, R be given. Set $Q = \pi^{-1}; R; \pi$. Define r by letting $a\ r\ b$ iff $a^1\ Q\ b^1$. We shall show that Q satisfies the condition in Theorem 1, so $Q \subseteq \bar{r}$, proving the lemma.
The abort condition states that $\gg; R \subseteq R; \gg$. So $\pi^{-1}; \gg; R; \pi \subseteq \pi^{-1}; R; \gg; \pi$. Since $\gg; \pi = \pi; \gg$ and $\gg; \pi^{-1} = \pi^{-1}; \gg$, we conclude that $\gg; Q \subseteq Q; \gg$. Choose A, B such that $A\ Q\ B$. Note that A and B contain labeled nodes only. Let $a \in A$. From the abort condition above, by repeatedly removing all nodes but a, we find $a^1 \leq A, Y \leq B$ such that $a^1\ R\ Y$. By the trace condition, since $a^1 \rhd^{\ell(a^1)} a^1$, $Y = b^1$ for some $b \in B$ with $\ell(a) = \ell(b)$, and thus $a\ r\ b$. Again from the abort condition, there must exist an $A' \leq A$ such that $A'\ Q\ (B - b)$. By the trace condition, this implies the existence of a $c \in A$ such that $A' = A - c$. So we can indeed apply Corollary 1. This completes our proof. □

4.3 Congruence properties

In this part, we prove that SE bisimilarity is a congruence for several operators. We define the free merge, relabeling, synchronization and split (decompose) operators. Other operators (like action prefixing) can be treated too. SE bisimilarity is not a congruence for the event-oriented (ACP,CCS) choice operators; a strengthening (root condition) is required to achieve this.
Merging two nets or processes is nothing but juxtaposing them, resulting in disjoint components. By *synchronizing* between these components they can become connected. Relabeling a net means replacing every true label a by its function value $\phi(a)$ for a given relabeling function ϕ. By relabeling with τ ("unlabeling"), one achieves abstraction. Synchronizing a net means the fusion of transitions with prescribed true labels. Splitting means decomposing transitions with certain labels into a sequence of two transitions, giving three nodes with new labels. In the formal definitions below, note that different nets can always be made disjoint modulo isomorphism.

Definition 7 *Let $L = (P, T, I, O, \ell)$ and $M = (P', T', I', O', \ell')$ be nets with disjoint nodes. Let $E \in \mathcal{B}(P_L); F \in \mathcal{B}(P_M)$.*

Free Merge *The free merge $L||M$ of L and M is the net $(P \cup P', T \cup T', I \cup I', O \cup O', \ell \cup \ell')$. The free merge $(L, E)||(M, F)$ of the processes (L, E) and (M, F) is defined by $(L, E)||(M, F) = (L||M, E \cup F)$.*

Relabeling *Let $\phi \in \mathcal{L}_\tau \to \mathcal{L}_\tau$, with $\phi(\tau) = \tau$. The ϕ place (transition) relabeling L^p_ϕ (L^t_ϕ) of L is obtained by replacing every true place (transition) label x by $\phi(x)$.*

Synchronization *Let $\chi \in \mathcal{L}^2 \hookrightarrow \mathcal{L}_\tau$ such that $\chi(a, b) = \chi(b, a)$ The χ-synchronization $L^s_\chi(L)$ of L is obtained by fusion of any two transitions t, s with $(\ell(t), \ell(s)) \in \mathrm{dom}(\chi)$ and labeling the fusion result $\mathbf{f}(t, s)$ with $\chi(\ell(t), \ell(s))$.*

Decomposition *Let $\psi \in \mathcal{L} \hookrightarrow \mathcal{L}^3_\tau$. The ψ-decomposition $L^d_\psi(L)$ of L is obtained by replacing any transition t with $\ell(t) \in \mathrm{dom}(\psi)$ by two transitions $\mathbf{i}(t), \mathbf{o}(t)$ connected via a place $\mathbf{s}(t)$. The input arcs of t are connected to $\mathbf{i}(t)$ and its output arcs to $\mathbf{o}(t)$.*

L	L^r_ϕ	L^s_χ	L^d_ψ
P	P	P	$P \cup \mathrm{ran}(\mathbf{s})$
T	T	$(T - \mathrm{dom}(\mathbf{f})) \cup \mathrm{ran}(\mathbf{f})$	$(T - \mathrm{dom}(\mathbf{i}) \cup \mathrm{ran}(\mathbf{i}) \cup \mathrm{ran}(\mathbf{o})$
$I(x)$	$I(x)$	$I(x)$ if $x \in T$ $I(y) + I(z)$ if $x = \mathbf{f}(y,z)$	$I(x)$ if $x \in T$ $I(y))$ if $x = \mathbf{i}(y)$ $\mathbf{s}(y)$ if $x = \mathbf{o}(y)$
$O(x)$	$O(x)$	$O(x)$ if $x \in T$ $O(y) + O(z)$ if $x = \mathbf{f}(y,z)$	$O(x)$ if $x \in T$ $O(y)$ if $x = \mathbf{o}(y)$ $\mathbf{s}(y)$ if $x = \mathbf{i}(y)$
$\ell(x)$	$\phi(\ell(x))$	$\ell(x)$ if $x \in P \cup T$ $\chi(\ell(y), \ell(z))$ if $x = \mathbf{f}(y,z)$	$\ell(x)$ if $x \in P \cup T$ $\psi_i(\ell(x))$ if $x \in \mathrm{ran}(\mathbf{i})$ $\psi_s(\ell(x))$ if $x \in \mathrm{ran}(\mathbf{s})$ $\psi_o(\ell(x))$ if $x \in \mathrm{ran}(\mathbf{o})$

Table 3: Definition of operators on nets

Let Φ be a relabeling/synchronization/decomposition operator for nets. The same operator for processes is then defined by $\Phi(L, E) = (\Phi(L), E)$.
The formal constructions are given in Table 3. In that table, the following auxiliary definitions have been introduced. The function $\mathbf{f} \in T^2 \hookrightarrow \mathcal{N}$ is chosen such that $(t,s) \in \mathrm{dom}(\mathbf{f})$ iff $(\ell(t), \ell(s)) \in \mathrm{dom}(\chi)$ and $\mathrm{ran}(\mathbf{f}) \cap N = \emptyset$ and $\mathbf{f}(t,s) = \mathbf{f}(t',s') \Leftrightarrow \{t,s\} = \{t',s'\}$. The injections $\mathbf{i}, \mathbf{s}, \mathbf{o}$ in $T \hookrightarrow \mathcal{N}$ are chosen such that $\mathrm{dom}(\mathbf{i}) = \mathrm{dom}(\mathbf{s}) = \mathrm{dom}(\mathbf{o}) = \{t \in T \mid \ell(t) \in \mathrm{dom}(\psi)\}$ and $\mathrm{ran}(\mathbf{i}), \mathrm{ran}(\mathbf{s}), \mathrm{ran}(\mathbf{o}), P, T$ are mutually disjoint. The functions ψ_i, ψ_s, ψ_o are such that $\psi(x) = (\psi_i(x), \psi_s(x), \psi_o(x))$ for all $x \in \mathrm{dom}(\psi)$. The I, O and ℓ functions are defined pointwise; their domains have been defined higher in the table. Note that the choice of nodes for $\mathbf{f}, \mathbf{i}, \mathbf{s}$ and $\mathbf{o}$ is irrelevant modulo isomorphism.

In Figure 10, the net in the middle is L, the net on the right is L^s_χ, where $\chi = \{((a,b),c)\}$ and the net on the left is L^d_ψ, where $\psi = \{(a,(c,\tau,d))\}$.
The split and synchronization operators are event-oriented. State-oriented place split and fusion operators can be defined as well. SE bisimilarity is not a congruence w.r.t. place fusion; some additional conditions are needed. We now proceed to show that (structural) SE bisimilarity is a congruence for the defined operators.

Theorem 3 *Let Φ be a given relabeling, synchronization or split operator. Let L, L' and M be nets such that $L \sim L'$. Then $L||M \sim L'||M$ and $\Phi(L) \sim \Phi(L')$. Let $(L,E), (L',E')$ and (M,F) be processes such that $(L,E) \sim (L',E')$. Then $((L,E)||(M,F)) \sim ((L',E')||(M,F))$ and $\Phi(L,E) \sim \Phi(L',E')$.*

Proof: Let R be a SE bisimulation between L and L'. Let r be the relation between the labeled nodes of L and L' from Lemma 3.
The elements of $\mathcal{B}(N_{L||M})$ can be written uniquely as a sum $X + Y$, where $X \in \mathcal{B}(N_L)$

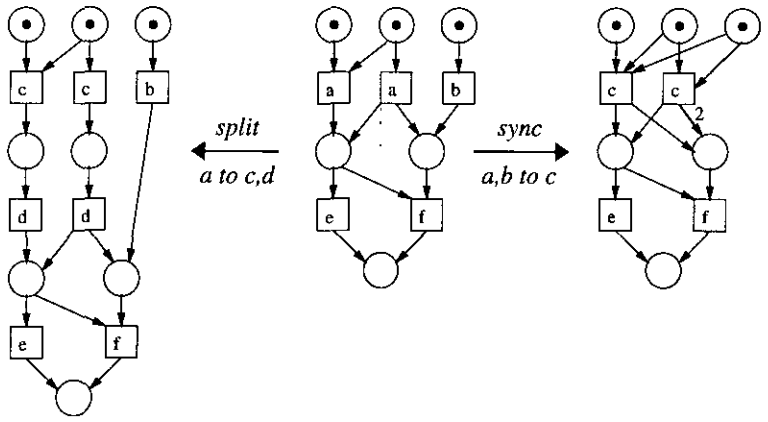

Figure 10: Synchronization and split examples

and $Y \in \mathcal{B}(N_M)$ and likewise for $\mathcal{B}(N_{L'||M})$. We define $R_1 \in \mathcal{P}(\mathcal{B}(N_{L||M}) \times \mathcal{B}(N_{L'||M}))$ by $X + Y \; R_1 \; X' + Y'$ iff $X \; R \; X'$ and $Y = Y'$.

If Φ is a relabeling operator, then we define $R_2 \in \mathcal{P}(\mathcal{B}(N_{\Phi(L)}) \times \mathcal{B}(N_{\Phi(L')}))$ by $R_2 = R$.

If Φ is a synchronization operator, the elements of $\mathcal{B}(N_{\Phi(L)})$ can be written uniquely as a sum $X + Y + Z$, where X is a place bag, Y a bag of unsynchronized transitions and Z a bag of synchronized transitions. So $X \in \mathcal{B}(P_L)$, $Y \in \mathcal{B}(T_L)$ and $Z \in \mathcal{B}(T_{\mathrm{ran}(\mathbf{f})})$. Note that Z stems from pairs of transitions labeled in L. We do likewise for $\Phi(L')$, with $\mathbf{f}$ replaced by $\mathbf{f}$'. We define $R_3 \in \mathcal{P}(\mathcal{B}(N_{\Phi(L)}) \times \mathcal{B}(N_{\Phi(L')}))$ by $X + Y + Z \; R_3 \; X' + Y' + Z'$ iff $X \; R \; X'$ and $Y \; R \; Y'$ and $Z \; \bar{s} \; Z'$, where $\mathbf{f}(x, y) \; s \; \mathbf{f}'(z, w)$ iff $x \; r \; z$ and $y \; r \; w$.

If Φ is a split operator, the elements of $\mathcal{B}(N_{\Phi(L)})$ can be written uniquely as a sum $X + Y + Z + W + V$, where X is a bag of old places, Y of unsplit transitions, Z of start transitions, W of new places and V of finish transitions. So $X \in \mathcal{B}(P_L)$, $Y \in \mathcal{B}(T_L)$, $Z \in \mathcal{B}(\{\mathbf{i}(x) \mid x \in T_L\})$, $W \in \mathcal{B}(\{\mathbf{s}(x) \mid x \in T_L\})$ and $V \in \mathcal{B}(\{\mathbf{o}(x) \mid x \in T_L\})$. Note that Z, W and V stem from transitions labeled in L. We do likewise for $\Phi(L')$. We define $R_4 \in \mathcal{P}(\mathcal{B}(N_{\Phi(L)}) \times \mathcal{B}(N_{\Phi(L')}))$ by $X + Y + Z + W + V \; R_4 \; X' + Y' + Z' + W' + V'$ iff $X \; R \; X'$ and $Y \; R \; Y'$, $Z \; \bar{s} \; Z'$, $W \; \bar{t} \; W'$ and $V \; \bar{u} \; V'$, where $x \; s \; y$ iff $\mathbf{i}^{-1}(x) \; r \; \mathbf{i}^{-1}(y)$, $x \; t \; y$ iff $\mathbf{s}^{-1}(x) \; r \; \mathbf{s}^{-1}(y)$, $x \; u \; y$ iff $\mathbf{o}^{-1}(x) \; r \; \mathbf{o}^{-1}(y)$.

The relations R_i are SE bisimulations for the new nets. Its proof is tedious, but straightforward. If R is total and surjective, the same holds for the R_i. If $E \; R \; E'$ then also $E \; R_i \; E'$ for all i. □

The above theorem depends upon the abort condition, as the relation r from Lemma 3 is essential in the construction of the new bisimulations.

4.4 Preservation of branching structure

We now prove that SE bisimilarity is stronger than (interleaved) branching bisimilarity. In [10], branching bisimilarity is characterized as the coarsest equivalence relation that fully

preserves the branching structure of a process. Branching bisimilarity is event-oriented, so we disregard places in nets. We adopt the definition from [4], which is an improvement over earlier versions. The definition in [13] is closely related. These definitions (and older ones) have different notions of branching bisimulation, but the same notion of branching bisimilarity [4].
We recall the definitions of $\xrightarrow{a}$ for $a \in \mathcal{L}$ and $\xrightarrow{\tau}$ for place bags from Definition 5 and add some related notions.

Definition 8 *Let L be a net, $\alpha \in \mathcal{L}_\tau$ and $X, Y \in \mathcal{B}(P)$.*
We set $X \xrightarrow{\alpha} Y$ iff there exists a $t \in T$ with $\ell(t) = \alpha$ and $Z \in \mathcal{B}(P)$ such that $X\ s\ (t + Z)$ and $(t + Z)\ c\ Y$.
Let $\Longrightarrow$ be the reflexive-transitive closure (zero or more silent steps) of $\xrightarrow{\tau}$ defined by $\Longrightarrow = D_{\mathcal{B}(P)} \cup (\xrightarrow{\tau})^+$. We set $\xrightarrow{(a)} = \xrightarrow{a}$ and $\xrightarrow{(\tau)} = D_{\mathcal{B}(P)} \cup (\xrightarrow{\tau})$ (zero or one silent step).

The set $\mathcal{B}(P)$ and the relations $\xrightarrow{\alpha}$ constitute the *process space* related to the net L.

Definition 9 *An* η-simulation *between nets L and M is a relation $R \in \mathcal{P}(\mathcal{B}(P_L) \times \mathcal{B}(P_M))$ such that for each x, x', y with $x\ R\ y$ and $x \xrightarrow{\alpha} x'$ (with $\alpha \in \mathcal{L}_\tau$) there exist y'', y' such that $y \Longrightarrow y'' \xrightarrow{(\alpha)} y'$ and $x\ R\ y''$ and $x'\ R\ y'$.*
A branching bisimulation *is an η-simulation the inverse of which is also an η-simulation. Processes (L, E) and (M, F) are* branching bisimilar *iff there exists a branching bisimulation R between L and M such that $E\ R\ F$.*

We shall prove that an SE bisimulation is a branching bisimulation. The proof is rather technical, so we sketch its idea. An illustration can be found in Figure 11. Supposing an SE bisimulation R between L and M, we let x, y, x' be place-only states such that $x\ R\ y$ and $x \xrightarrow{\alpha} x'$. From Definition 8, we find a state x_t between x and x'. By the trace condition of SE bisimilarity, we find a state $A + U$ of M, where A is a place bag and U a transition bag. From $A + U$ we can reach $A + I(U)$ via ρ and $A + O(U)$ via γ. By the rollback and commit conditions of SE bisimilarity, we can deduce that $x\ R\ (A + I(U))$ and $x'\ R\ (A + O(U))$. We then prove that there must be one transition u in U that performs the essential step from a state related to x to a state related to x'. From it, we derive y'' and y' satisfying the η-simulation requirements.
Before proving the theorem, we'll have some lemma's.

Lemma 4 *Let x be a place-only state of a net L and let y be a state such that $x\ (s \cup c)^k\ y$. Then there exists a place-only state y' and numbers l, m such that $y\ (s; c)^l\ y'$ and $y'\ s^m\ y$.*

Proof: We use induction w.r.t. k. If $k = 0$, we take $y' = y$ and $l = m = 0$. So let $k > 0$. Then there exists a z such that $x\ (s \cup c)^{k-1}\ z\ (s \cup c)\ y$. By the induction hypothesis, there exists a z' and numbers l', m' such that $x\ (s; c)^{l'}\ z'$ and $z'\ s^{m'}\ z$. If $z\ s\ y$, then we can take $l = l'$, $y' = z'$ and $m = m' + 1$. If $z\ c\ y$, then $m' > 0$, since z' is place-only. We use induction on $m' - 1$ and the inclusion $s; s; c \subseteq s; c; s$ of Lemma 1 to prove that $z'\ (s; c; s^{m'-1})\ y$.

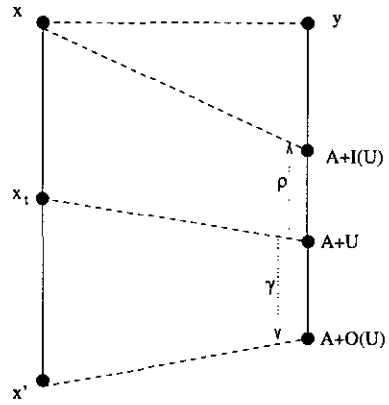

Figure 11: Proof of Theorem 4

So there exists a y' such that $z\ (s;c)\ y'\ s^{m'-1}\ ys$ and we can take $l = l'+1$ and $m = m'-1$. □

Another lemma gives a property of SE bisimilarity w.r.t. place-only states. For any state $A+T$ of a net L, where A is a place bag and T a transition bag, we define $\hat{\rho}(A+T) = A + I(T)$ and $\hat{\gamma}(A+T) = A + O(T)$. Here I, O are lifted to transition bags by setting $I(\Sigma_{i\in J} t_i^1) = \Sigma_{i\in J} I(t_i)$ and $O(\Sigma_{i\in J} t_i^1) = \Sigma_{i\in J} O(t_i)$. $\hat{\rho}(X)$ is the state reached from X by revoking all transitions pending in X and $\hat{\gamma}(X)$ is the state reached from X by committing them.

Lemma 5 *Let L, M be nets and R an SE bisimulation between them. Then $X\ R\ Y$ implies $\hat{\rho}(X)\ R\ \hat{\rho}(Y)$ and $\hat{\gamma}(X)\ R\ \hat{\gamma}(Y)$.*

Proof: Let $X = A + T$ and $Y = B + U$ where A, B are place bags and T, U transition bags. We have that $X\ \rho\ \hat{\rho}(X) = A + I(T)$, so by the rollback condition, there exists a Z such that $Y\ \rho\ Z$ and $\hat{\rho}(X)\ R\ Z$. Now $Z\ \rho\ \hat{\rho}(Z)$, so again there exists a W such that $\hat{\rho}(X)\ \rho\ W$ and $W\ R\ \hat{\rho}(Z)$. Since $\hat{\rho}(X)$ is place-only, $W = \hat{\rho}(X)$. Also $\hat{\rho}(Z) = \hat{\rho}(Y)$. This proves the $\hat{\rho}$ part. The $\hat{\gamma}$ part is analogous, using the commit condition. □

Theorem 4 *Let L, M be nets and E, F initial states such that $(L, E) \sim (M, F)$. Then (L, E) and (M, F) are branching bisimilar.*

Proof: We may assume wlog that L, M have unlabeled places. If not, we relabel all places with τ. The new processes are still SE bisimilar, whereas the definition of branching bisimilarity does not depend upon the place labels.

Let R be an SE bisimulation between L and M such that $E\ R\ F$. We shall show that R and its inverse satisfy the conditions in Definition 9, when restricted to place-only states. By symmetry, we only need to prove it for R.

Now suppose that x, x' are place-only states of L and y of M such that $x\ R\ y$ and $x \xrightarrow{\alpha} x'$. We shall prove that there exist states y'', y' such that $x\ R\ y''$ and $x'\ R\ y'$ such that $y \Longrightarrow y''$ and $y'' \xrightarrow{(\alpha)} y'$.

Since $x \xrightarrow{\alpha} x'$, there exist a transition t with $\ell(t) = \alpha$ such that $x' = x - I(t) + O(t)$ and $x\ s\ x_t\ c\ x'$, where $x_t = x - I(t) + t$. Since places are unlabeled, we have $x \rhd_L^\sigma x_t$, where $\sigma = \langle \mathbf{0} \rangle$ if $\alpha = \tau$ and $\langle \mathbf{0}, \alpha \rangle$ otherwise. By the trace condition, there must exist a state $A + U$, where A is a place bag and U a transition bag, such that $y \rhd_M^\sigma A + U$ and $x_t\ R\ A + U$. We may assume that U is the smallest bag with this property.

By the structure of σ, all transitions in U have label τ except at most one, which has label α. All other transitions that are started and committed in the sequence between y and $A + U$ have label τ too. By Lemma 4, we can find y_i's and rearrange the events between y and $A + U$ such that $y\ (s;c)^l\ A + I(U) = y_0\ s\ \ldots\ s\ y_n = A + U$. From Definition 8 and the fact that y and $A + I(U)$ are place-only states we conclude that $y \Longrightarrow A + I(U)$. Since U was the smallest bag, we either have that $n = 0$ or not $x_t\ R\ y_{n-1}$.

If $n = 0$, we have that $\alpha = \tau$, $U = \mathbf{0}$, $x_t\ R\ A$, so by Lemma 5 since $\hat{\rho}(x_t) = x, \hat{\gamma}(x_t) = x'$ and $\hat{\rho}(A) = \hat{\gamma}(A) = A$, $x\ R\ A$ and $x'\ R\ A$. We take $y'' = y' = A$ and observe that $y \Longrightarrow y''$ and $y'' \xrightarrow{(\alpha)} y'$.

So let $n > 0$. There exists a transition $u \in U$ such that $y_n = y_{n-1} - I(u) + u^1$ and $\ell(u) = \alpha$. Since $y_n\ \rho\ y_{n-1}$, there must exist a ξ such that $x_t\ \rho\ \xi$ and $\xi\ R\ y_{n-1}$. Since $x_t\ R\ y_{n-1}$ was false, we conclude that $\xi = x$. Now let $y'' = \hat{\gamma}(y_{n-1})$ and $y' = \hat{\gamma}(y_n)$. We have that $y''\ s\ y'' - I(u) + u^1\ c\ y'$, so $y'' \xrightarrow{\alpha} y'$. By Lemma 5, $y''\ R\ \hat{\gamma}(x) = x$ and $y'\ R\ \hat{\gamma}(x_t) = x'$. This concludes our proof. □

The proof uses the trace, commit and rollback conditions. Of course the trace condition is essential. The nets C and D in Figure 8 show that the rollback condition is necessary, since these nets are not branching bisimilar.

4.5 Reduction of LPT nets

In this paper we do not intend to formally define rules that allow us to transform nets modulo SE bisimilarity. We just give a few examples of such rules. Reduction rules that are *local* are preferred. They take into consideration only a small portion of the net and simplify the net by diminishing the number of nodes and/or edges.

From [8] one may deduce that it is unlikely that processes can be algorithmically reduced to a normal from modulo SE bisimilarity. However it seems possible to develop algorithms that reduce nets to a normal form modulo *structural* SE bisimilarity.

In this paper we give examples of a few local reduction rules that allow us to verify our burger example. We treat the *merge*, *removal* and *short-circuit* rules.

The merge rule is the same as for place bisimilarity. Places and transitions having the same effect can be merged, like the nets C, D in Figure 1. In most cases, it can be decided locally whether this merge is possible. A global algorithm as in [2] exists for computing the "maximal" merge. The special nature of unlabeled places or transitions is not taken into account by this rule.

The removal rule removes *redundant* τ-labeled nodes. A place is redundant if adding

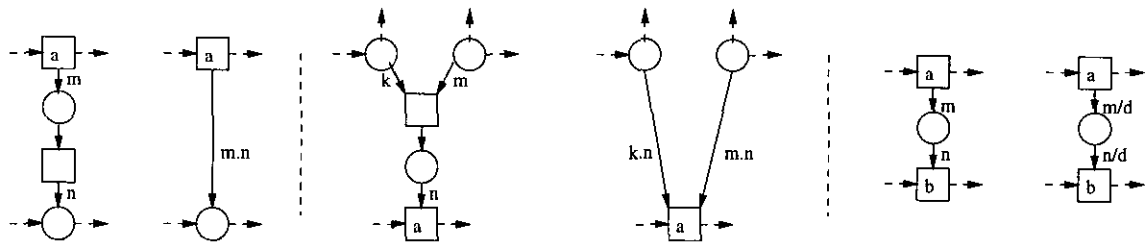

Figure 12: Short circuit / weight recalculation examples

tokens to it will not allow new future events. A transition is redundant iff it is connected to redundant places. The reduction removes the node and edges leading to and from it. Again, redundancy is a global property that can be decided locally in most cases. An illustration is given by the nets in Figure 3. The place in net A is redundant, since it can become filled with tokens without any visible witness of the fact. Adding tokens would not influence the process. Hence the unlabeled transition is redundant too. Another redundant place occurs in net C; although it enables a visible transition, the existence of the other b-labeled transition makes it redundant.

The short-circuit rule applies to τ-labeled transitions without labeled pre- and post places. If this does not affect the branching structure of the process, the pre-and post places can be merged, removing the node in between. In doing so, arc weights can be recalculated. This is illustrated in Figure 12, which contains also a pure weight recalculation rule. In this figure, n, k, m are positive integers representing arc weights and d is a common divisor of m and n. a and b are transition labels that may be absent. Dashed arrows indicate possible other input and output arcs.

We apply the above rules to our burger example. In the leftmost net all labels are different, so we use them to identify the nodes. If all nodes except `order`, `pay` and `burger` are abstracted from, the places to the left of `fry/assemble` become redundant, so we apply the removal rule. The transitions in between are removed too, by the same rule. This leaves us with a chain that can be short-circuited, giving the net in the upper right.

For the other verification, note that `ao`, `pay` and either one of the places `ub`, `lb` are redundant and can be removed (the other one then ceases to be redundant!). We can use the rules in Figure 12 to obtain the net in the lower right.

The fact that SE bisimilarity is a congruence for decomposition and synchronization makes it a congruence for action refinement of any kind. An example is given in Figure 13. On the left, we have a net containing the action `pay`. We may wish to refine this action as a choice between cash (`pca`) and credit card (`pcc`) payment. In case of cash payment a certain amount of change (`rch`) is received back. The figure shows how this can be achieved by the split, merge, synchronize and rename operators and then reducing the result.

From the fact the SE bisimilarity is a congruence for these operators, we conclude that it is a congruence for this refinement of the `pay` action. One need not redo the verification of the customer protocol in Figure 4 if this refinement is applied to both specification and

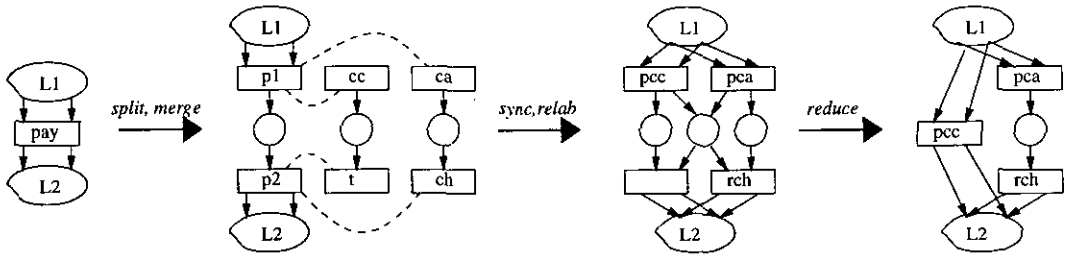

Figure 13: An example refinement

implementation.

5 Conclusions and further work

This paper introduces an equivalence relation for nets that supports both state-based and event-based modeling and verification and is a congruence for action refinement. It is thus possible to specify and implement a system on a high level, verify that the implementation is correct and then refine transitions and repeat the process on lower levels. It is not necessary to "flatten" the hierarchy. One does not need to adhere to the "durational" interpretation of events. Even when events can be considered atomic, the possibility of hierarchical verification can be appreciated. A hierarchical approach w.r.t. places is supported in the same way.

SE bisimilarity features four bisimulation conditions, trace, abort, commit and rollback. We shall discuss in what extent these conditions are necessary. Removal, weakening and strenghtening of the conditions all seem possible. This will result in different equivalence relations. There is no clear reason to prefer one over the other.

The trace condition is the most important one for preserving the branching structure of processes. One may argue that this condition is too restrictive. In [9], various weakenings are given that may be translated to the present framework. On the other hand, one may wish to detect *divergence*, the possibily of an infinite τ-loop. The $\rhd^\sigma$ relation can be refined to include the possible divergences in the trace σ. The trace condition with this new relation then gives rise to a divergence-sensitive SE bisimilarity.

The abort condition is vital for the hierarchical approach based on refinement. In Figure 14, we see a refinement that introduces a "deadlock" within an action a. This example makes it plausible that if two states containing a actions are related, there must also exist related states with one a action less, which is essentially the abort condition.

By the rollback condition, SE bisimilarity is a congruence for operators that revoke pending actions. This fact makes SE bisimilarity suitable for verifying workflow procedures. There exist many workflow engines using models that have a Petri net semantics. Tasks are modeled by transitions. In the workflow context, tasks do have a duration and may be revoked when started.

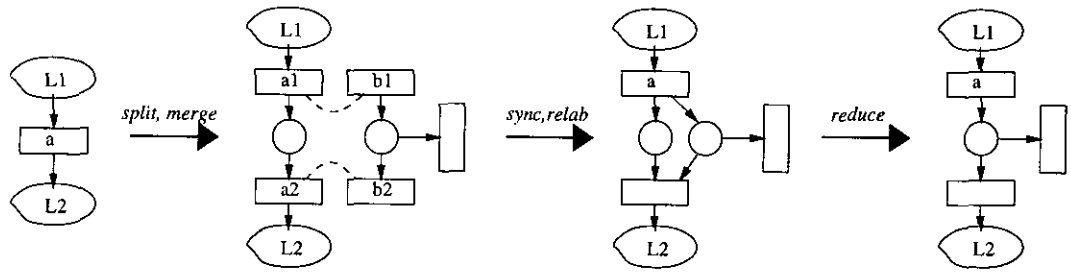

Figure 14: A deadlock refinement

As indicated earlier, the commit condition is probably redundant. A "weaker" variant of SE bisimilarity without the rollback and commit conditions seems well possible and may be related to weak (or delay) bisimilarity [13].

Research on SE bisimilarity should continue. Possible issues are the removal of the commit condition and the definition and properties of operators that are based on place labels. Most important is the development of algorithms to reduce a net to some normal form modulo structural SE bisimilarity. Such an algorithm, with additional reduction rules for processes, can be built in tools, allowing for improved verification of concurrent systems.

Acknowledgements

I wish to thank Twan Basten, Andries Brouwer, Michel Reniers, Jaap van der Woude and several referees of earlier versions of this paper for their help and advice.

References

1. C. Autant, W. Pfister, and Ph. Schnoebelen. Place Bisimulations for the Reduction of Labeled Petri Nets with Silent Moves. In *Proceedings International Conference on Computing and Information*, 1994. `http://phoenix.trentu.ca/jci/papers/icci94/A14/P001.html`.

2. C. Autant and Ph. Schnoebelen. Place Bisimulations in Petri Nets. In K. Jensen, editor, *Application and Theory of Petri Nets 1992, 13th. International Conference, Proceedings*, volume 616 of *Lecture Notes in Computer Science*, pages 45–61. Springer–Verlag, Berlin, Germany, 1992.

3. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1990.

4. T. Basten. Branching Bisimilarity is an Equivalence Indeed! *Information Processing Letters*, 58:141–147, 1996.

5. E. Best, R. Devillers, and J. Hall. The Petri Box Calculus: a New Causal Algebra with Multilabel Communication. In G. Rozenberg, editor, *Advances in Petri Nets 1992*, volume 609 of *Lecture Notes in Computer Science*, pages 21–69. Springer–Verlag, Berlin, Germany, 1992.

6. K.M. Chandy and J. Misra. *Parallel Program Design: a Foundation.* Addison-Wesley, Reading, USA, 1988.

7. R. De Nicola and F. Vaandrager. Action versus State Based Logics for Transition Systems. In I. Guessarian, editor, *Semantics of Systems of Concurrent Processes, LITP Spring School, Proceedings*, volume 469 of *Lecture Notes in Computer Science*, pages 407–419. Springer–Verlag, Berlin, Germany, 1990.

8. J. Esparza and M. Nielsen. Decidability Issues for Petri Nets – a Survey. *Journal of Information Processing and Cybernetics*, 30(4):143–160, 1994.

9. R.J. van Glabbeek. The Linear Time - Branching Time Spectrum II. In E. Best, editor, *Proceedings CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 66–81. Springer–Verlag, Berlin, Germany, 1993.

10. R.J. van Glabbeek. What is Branching Time Semantics and Why to Use it. *Bulletin of the EATCS*, 53:190–198, 1994.

11. R.J. van Glabbeek and U. Goltz. Equivalence Notions for Concurrent Systems and Refinement of Actions. In A. Kreczmar and G. Mirkowska, editors, *Mathematical Foundations of Computer Science 1989, 14th. International Symposium, Proceedings*, volume 379 of *Lecture Notes in Computer Science*, pages 237–248. Springer–Verlag, Berlin, Germany, 1989.

12. R.J. van Glabbeek and F. Vaandrager. Petri Net Models for Algebraic Theories of Concurrency. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *PARLE: Parallel Architectures and Languages 1987, Vol II: Parallel Languages*, volume 259 of *Lecture Notes in Computer Science*, pages 224–242, Eindhoven, Netherlands, 1987. Springer–Verlag, Berlin, Germany.

13. R.J. van Glabbeek and W.P. Weijland. Branching Time and Abstraction in Bisimulation Semantics. *Journal of the ACM*, 43(3):555–600, 1996.

14. M. Hall. *Combinatorial Theory.* Wiley, Chichester, UK, 1986.

15. K. Jensen. *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* EATCS monographs on Theoretical Computer Science. Springer–Verlag, Berlin, Germany, 1992.

16. Z. Manna and A. Pnueli. *The Temporal Logic of Reactive and Concurrent Systems, Specification.* Springer–Verlag, Berlin, Germany, 1991.

17. R. Milner. *Communication and Concurrency.* Prentice–Hall, London, UK, 1989.

18. L. Pomello, G. Rozenberg, and G. Simone. A Survey of Equivalence Notions for Net Based Systems. In G. Rozenberg, editor, *Advances in Petri Nets 1992*, volume 609 of *Lecture Notes in Computer Science*, pages 410–472. Springer–Verlag, Berlin, Germany, 1992.

19. W. Vogler. Bisimulation and Action Refinement. *Theoretical Computer Science*, 114(1):173–200, 1993.

20. M. Voorhoeve. Structural Petri Net Equivalence. Computing Science Reports 96/07, Eindhoven University of Technology, 1996.

21. K Voss. Interface as a Basic Concept for Systems Specification and Verification. In K. Voss, H.J. Genrich, and G. Rozenberg, editors, *Concurrency and Nets*, pages 585–604. Springer–Verlag, Berlin, Germany, 1987.

Computing Science Reports

Department of Mathematics and Computing Science
Eindhoven University of Technology

In this series appeared:

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.
96/07	M. Voorhoeve	Structural Petri Net Equivalence, p. 16.
96/08	A.T.M. Aerts, P.M.E. De Bra, J.T. de Munk	OODB Support for WWW Applications: Disclosing the internal structure of Hyperdocuments, p. 14.
96/09	F. Dignum, H. Weigand, E. Verharen	A Formal Specification of Deadlines using Dynamic Deontic Logic, p. 18.
96/10	R. Bloo, H. Geuvers	Explicit Substitution: on the Edge of Strong Normalisation, p. 13.
96/11	T. Laan	AUTOMATH and Pure Type Systems, p. 30.
96/12	F. Kamareddine and T. Laan	A Correspondence between Nuprl and the Ramified Theory of Types, p. 12.
96/13	T. Borghuis	Priorean Tense Logics in Modal Pure Type Systems, p. 61
96/14	S.H.J. Bos and M.A. Reniers	The I^2 C-bus in Discrete-Time Process Algebra, p. 25.
96/15	M.A. Reniers and J.J. Vereijken	Completeness in Discrete-Time Process Algebra, p. 139.
96/17	E. Boiten and P. Hoogendijk	Nested collections and polytypism, p. 11.
96/18	P.D.V. van der Stok	Real-Time Distributed Concurrency Control Algorithms with mixed time constraints, p. 71.
96/19	M.A. Reniers	Static Semantics of Message Sequence Charts, p. 71
96/20	L. Feijs	Algebraic Specification and Simulation of Lazy Functional Programs in a concurrent Environment, p. 27.
96/21	L. Bijlsma and R. Nederpelt	Predicate calculus: concepts and misconceptions, p. 26.
96/22	M.C.A. van de Graaf and G.J. Houben	Designing Effective Workflow Management Processes, p. 22.
96/23	W.M.P. van der Aalst	Structural Characterizations of sound workflow nets, p. 22.
96/24	M. Voorhoeve and W. van der Aalst	Conservative Adaption of Workflow, p.22
96/25	M. Vaccari and R.C. Backhouse	Deriving a systolic regular language recognizer, p. 28
97/01	B. Knaack and R. Gerth	A Discretisation Method for Asynchronous Timed Systems.
97/02	J. Hooman and O. v. Roosmalen	A Programming-Language Extension for Distributed Real-Time Systems, p. 50.
97/03	J. Blanco and A. v. Deursen	Basic Conditional Process Algebra, p. 20.
97/04	J.C.M. Baeten and J.A. Bergstra	Discrete Time Process Algebra: Absolute Time, Relative Time and Parametric Time, p. 26.
97/05	J.C.M. Baeten and J.J. Vereijken	Discrete-Time Process Algebra with Empty Process, p. 51.
97/06	M. Franssen	Tools for the Construction of Correct Programs: an Overview, p. 33.
97/07	J.C.M. Baeten and J.A. Bergstra	Bounded Stacks, Bags and Queues, p. 15.

97/08	P. Hoogendijk and R.C. Backhouse	When do datatypes commute? p. 35.
97/09	Proceedings of the Second International Workshop on Communication Modeling, Veldhoven, The Netherlands, 9-10 June, 1997.	Communication Modeling- The Language/Action Perspective, p. 147.
97/10	P.C.N. v. Gorp, E.J. Luit, D.K. Hammer E.H.L. Aarts	Distributed real-time systems: a survey of applications and a general design model, p. 31.
97/11	A. Engels, S. Mauw and M.A. Reniers	A Hierarchy of Communication Models for Message Sequence Charts, p. 30.
97/12	D. Hauschildt, E. Verbeek and W. van der Aalst	WOFLAN: A Petri-net-based Workflow Analyzer, p. 30.
97/13	W.M.P. van der Aalst	Exploring the Process Dimension of Workflow Management, p. 56.
97/14	J.F. Groote, F. Monin and J. Springintveld	A computer checked algebraic verification of a distributed summation algorithm, p. 28
97/15	M. Franssen	λP-: A Pure Type System for First Order Loginc with Automated Theorem Proving, p.35.
97/16	W.M.P. van der Aalst	On the verification of Inter-organizational workflows, p. 23
97/17	M. Vaccari and R.C. Backhouse	Calculating a Round-Robin Scheduler, p. 23.
97/18	Werkgemeenschap Informatiewetenschap redactie: P.M.E. De Bra	Informatiewetenschap 1997 Wetenschappelijke bijdragen aan de Vijfde Interdisciplinaire Conferentie Informatiewetenschap, p. 60.
98/01	W. Van der Aalst	Formalization and Verification of Event-driven Process Chains, p. 26.