Partial S-invariants for the verification of infinite systems families

Walter Vogler

Partial S-Invariants for the Verification of Infinite Systems Families

Walter Vogler

Report 2001-4

März 2001

Partial S-Invariants for the Verification of Infinite Systems Families

Walter Vogler

Institut für Informatik, Universität Augsburg
D-86135 Augsburg, Germany
email: vogler@informatik.uni-augsburg.de

Abstract

We introduce partial S-invariants of Petri nets, which can help to determine invariants and to prove safety if large nets are built from smaller ones using parallel composition with synchronous communication. We show how partial S-invariants can support compositional reduction and, in particular, a specific form of it called the fixed-point approach. With the latter, infinite parameterized families of concurrent systems can be verified. Partial S-invariants and the fixed-point approach are used to prove the correctness of two solutions to the MUTEX-problem based on token rings; for this, we only have to prove liveness of a simplified version due to previous results.

1 Introduction

For the verification of infinite parameterized families of concurrent systems the so-called behavioural fixed-point approach is advocated in [VK98]. The members of such families are composed of an increasing number of components, this number being the parameter. If one can show that the composition of, say, two of these components is equivalent to just one component, then one can reduce each member of the family to an equivalent small one, and it suffices to prove this small system to be correct. This approach is a specific case of compositional reduction, for which it is essential that the equivalence under consideration is a congruence for composition – and, of course, it must be strong enough to support verification. We will model systems with Petri nets, and we will use parallel composition with synchronization of common actions, which corresponds to merging transitions; also renaming and hiding of actions are important.

In this paper, we will apply the fixed-point approach to two token-ring based solutions for the problem of mutual exclusion (MUTEX). Such a ring has a component for each user that needs access to the shared resource, and each component has a separate interface that allows the respective user to *request* access, *enter* its critical section – in which the resource is used –, and *leave* this section again. To verify such a ring, one has to show the safety property that there are never two users in their respective critical sections at the same time, i.e. that enter- and leave-actions alternate properly, and one has to show the liveness property that (essentially) to each requesting user access is granted eventually. Modelling a token-ring with Petri nets, MUTEX-safety is usually easy to show applying an S-invariant. Hence, we want to apply the fixed-point approach to prove MUTEX-liveness.

An immediate problem is that each ring component has an external interface to its user with actions of its own, and therefore we cannot expect that two components are equivalent to one. In [BV98], we have shown that under some symmetry assumption it is sufficient to check MUTEX-liveness for one user and hide the other user-actions from the interface. In the modified net, only the actions of one user are visible and two components may be equivalent to one; hence, and this is the first point to be made, our symmetry result opens the door for applying the fixed-point approach.

In fact, one encounters another problem: the composition of two components may not be equivalent to one, because in isolation these nets exhibit behaviour that is not possible in the complete ring. To show the equivalence, one has to restrict their behaviour in a suitable way; this is somewhat similar e.g. to the interface descriptions presented in [GSL96]. The main contribution of this paper is the development of what we call *partial S-invariants* in order to restrict the behaviour suitably. We show how partial S-invariants can support compositional reduction in general, and we will apply them specifically in the fixed-point approach.

Partial S-invariants of components can also be used to obtain S-invariants of composed systems. Another notion of partial S-invariants – for a setting where nets are composed by merging places – has been defined in [Kin95] where it has also been shown how to combine these to obtain S-invariants of composed systems.

The equivalence we use is based on fairness in the sense of the progress assumption, i.e. weak fairness. In [Vog97], one can find such a semantics that is compositional for safe nets. Here, we have to deal with components of a safe net that are not safe themselves, and we show that compositionality for general nets can be achieved very similarly to [Vog97], if one uses a suitable generalization of weak fairness from safe to general S/T-nets. Since we are really interested in safe nets, we are free to choose any generalization that is convenient.

The Petri nets of this paper may have so-called read arcs, which are somewhat similar to loops. If transition t and place s form a loop, then firing t removes a token from s and returns it at the end; hence, this token is not available while t is firing. If t and s are connected by a read arc instead, then t checks for a token on s without actually using it; thus, other transitions might do the same while t is firing. For example, read arcs can model the concurrent reading of data. When we consider firing sequences only, read arcs and loops are interchangeable; when we consider concurrent behaviour or the progress assumption, they make a difference. It is shown in [Vog97] that ordinary nets without read arcs cannot solve the MUTEX-problem. Read arcs have found quite some interest recently, see e.g. [MR95, JK95, VSY98, BBCP00], and we include them for generality (in particular in the treatment of fairness) and because we need them in our applications.

Section 2 defines Petri nets with read arcs and the operations on nets we will use to construct the MUTEX-solutions. Section 3 gives our definition of (weak) fairness in the sense of progress assumption, refines the resulting semantics to a precongruence for our operations and shows full abstraction. Section 4 introduces partial S-invariants, shows how to combine them to S-invariants and presents the essential result for applying them for compositional reduction and, hence, in the fixed-point approach. Section 5 quotes from [BV98] the correctness definition for MUTEX-solutions and the symmetry result mentioned above. Section 6 shows how to use partial S-invariants in the fixed-point approach and proves two families of nets correct. For the second family, we use the tool FastAsy that compares the performance of asynchronous systems; the respective performance preorder is closely related to the precongruence we use in the present paper. We close with a discussion of related work in Section 7.

2 Basic Notions and Operations for Petri Nets with Read Arcs

In this section, we introduce Petri nets, which are extended with read arcs as explained in the introduction, and the basic firing rule. Then we define parallel composition, renaming and hiding for such nets and give some laws for these operations. For general information on ordinary Petri nets, the reader is referred to e.g. [Pet81, Rei85]. The transitions of our nets are labelled with actions from some infinite alphabet Σ or with the empty word λ. In general, these actions are left uninterpreted; the labelling only indicates that two transitions with the same label from Σ represent the same action occurring in different internal situations, while λ-labelled transitions represent internal, unobservable actions.

Thus, a *labelled Petri net with read arcs* $N = (S, T, F, R, l, M_N)$ (or just a *net* for short) consists of finite disjoint sets S of *places* and T of *transitions*, the *flow relation* $F \subseteq S \times T \cup T \times S$ consisting of (ordinary) *arcs*, the set of *read arcs* $R \subseteq S \times T$, the *labelling* $l : T \to \Sigma \cup \{\lambda\}$, and the *initial marking* $M_N : S \to \mathbb{N}_0$, where $\mathbb{N}_0$ denotes the natural numbers including 0; we require that $(R \cup R^{-1}) \cap F = \emptyset$. When we introduce a net N or N_1 etc., then we assume that implicitly this introduces its components S, T, F, ... or S_1, T_1, ..., etc. The net is called *ordinary*, if $R = \emptyset$.

As usual, we draw transitions as boxes, places as circles and arcs as arrows; read arcs are drawn as lines (sometimes dashed) without arrow heads. As usual, nets N_1 and N_2 are *isomorphic*, written $N_1 = N_2$, if there is some function that bijectively maps the places of N_1 to the places of N_2 and the transitions of N_1 to the transitions of N_2 such that arcs, read arcs, labelling and initial marking are preserved. The *alphabet* $\alpha(N)$ of a net N is the set of all actions from Σ that occur as labels in N.

For each $x \in S \cup T$, the *preset* of x is ${}^\bullet x = \{y \mid (y, x) \in F\}$, the *postset* of x is $x^\bullet = \{y \mid (x, y) \in F\}$, and the *read set* of x is $\hat{x} = \{y \mid (y, x) \in R \cup R^{-1}\}$. If $x \in {}^\bullet y \cap y^\bullet$, then x and y form a *loop*. A *marking* is a function $S \to \mathbb{N}_0$ giving for each place a number of *tokens* – drawn as dots in the respective circle. We sometimes regard sets as characteristic functions, which map the elements of the sets to 1 and are 0 everywhere else; hence, we can e.g. add a marking and a postset of a transition or compare them componentwise.

We now define the basic firing rule, which extends the firing rule for ordinary nets by regarding a read arc (s, t) as loop, i.e. as ordinary arcs (s, t) and (t, s).

- A transition t is *enabled* under a marking M, denoted by $M[t\rangle$, if ${}^\bullet t \cup \hat{t} \leq M$.

If $M[t\rangle$ and $M' = M + t^\bullet - {}^\bullet t$, then we denote this by $M[t\rangle M'$ and say that t can *occur* or *fire* under M yielding the marking M'.

- This definition of enabling and occurrence can be extended to sequences as usual: a finite sequence w of transitions is *enabled* under a marking M, denoted by $M[w\rangle$, and yields the follower marking M' when *occurring*, denoted by $M[w\rangle M'$, if $w = \lambda$ and $M = M'$ or $w = w't$, $M[w'\rangle M''$ and $M''[t\rangle M'$ for some marking M'' and transition t. An infinite sequence w of transitions is *enabled* under a marking M, denoted as above, if all its finite prefixes are enabled under M. We denote the set of finite sequences over a set X by X^*, the set of infinite sequences by X^ω, and their union by X^∞. If $w \in T^\infty$ is enabled under the initial marking, then it is called a *firing sequence.*

- We can extend the labelling to sequences of transitions as usual, i.e. homomorphically; note that internal actions are automatically deleted in this *image* of a sequence. With this, we lift the enabledness and firing definitions to the level of actions: a sequence v of actions from Σ is *enabled* under a marking M, denoted by $M[v\rangle\rangle$, if there is some transition sequence w with $M[w\rangle$ and $l(w) = v$; for finite v, $M[v\rangle\rangle M'$ is defined analogously. If $M = M_N$, then v is called a *trace.*

- A marking M is called *reachable* if $M_N[w\rangle M$ for some $w \in T^*$. The net is *safe* if $M(s) \leq 1$ for all places s and reachable markings M and if all transitions t satisfy ${}^\bullet t \neq \emptyset$.

We are mainly interested in safe nets where the second condition is maybe non-standard, but also no serious restriction, since it can be satisfied by adding a loop between t and a new marked place if ${}^\bullet t$ were empty otherwise; this addition does not change the firing sequences. Since we will construct safe nets from components that, considered in isolation, violate one or both of the required conditions, we develop our approach for general nets.

Safe nets are a particular case of nets without self-concurrency: A transition t is *enabled self-concurrently* under a marking M, if ${}^\bullet t \cup \hat{t} \leq M - {}^\bullet t$, i.e. if there are enough tokens to enable two copies of t at the same time. A net is *without self-concurrency*, if, for all transitions t and reachable markings M, t is not enabled self-concurrently under M.

Next, we introduce parallel composition $\|$ where synchronization is over common actions. This is not much different from TCSP-like composition used in [Vog97], but makes notation lighter, and it is also used in [VK98]. If nets N_1 and N_2 with $A = \alpha(N_1) \cap \alpha(N_2)$ are combined using $\|$, then they run in parallel and have to synchronize on actions from A. To construct the composed net, we have to combine each a-labelled transition t_1 of N_1 with each a-labelled transition t_2 from N_2 if $a \in A$.

In the formal definition of parallel composition, $*$ is used as a dummy element, which is formally combined e.g. with those transitions that do not have their label in the synchronization set A. (We assume that $*$ is not a transition or a place of any net.) The *parallel composition* $N = N_1 \parallel N_2$ is defined by

$$\begin{array}{rcl}
S & = & S_1 \times \{*\} \cup \{*\} \times S_2 \\
T & = & \{(t_1,t_2) \mid t_1 \in T_1, t_2 \in T_2, l_1(t_1) = l_2(t_2) \in A\} \\
& & \cup\{(t_1,*) \mid t_1 \in T_1, l_1(t_1) \notin A\} \\
& & \cup\{(*,t_2) \mid t_2 \in T_2, l_2(t_2) \notin A\}
\end{array}$$

$$((s_1,s_2),(t_1,t_2)) \in F \quad \text{if} \quad \begin{cases} (s_1,t_1) \in F_1,\ s_1 \in S_1,\ t_1 \in T_1 \\ \text{or} \\ (s_2,t_2) \in F_2,\ s_2 \in S_2,\ t_2 \in T_2 \end{cases}$$

$$((t_1,t_2),(s_1,s_2)) \in F \quad \text{if} \quad \begin{cases} (t_1,s_1) \in F_1,\ s_1 \in S_1,\ t_1 \in T_1 \\ \text{or} \\ (t_2,s_2) \in F_2,\ s_2 \in S_2,\ t_2 \in T_2 \end{cases}$$

$$((s_1,s_2),(t_1,t_2)) \in R \quad \text{if} \quad \begin{cases} (s_1,t_1) \in R_1,\ s_1 \in S_1,\ t_1 \in T_1 \\ \text{or} \\ (s_2,t_2) \in R_2,\ s_2 \in S_2,\ t_2 \in T_2 \end{cases}$$

$$\begin{array}{rcl}
l((t_1,t_2)) & = & \begin{cases} l_1(t_1) & \text{if } t_1 \in T_1 \\ l_2(t_2) & \text{if } t_2 \in T_2 \end{cases} \\
M_N & = & M_{N_1} \dot\cup M_{N_2},\ \text{i.e. } M_N((s_1,s_2)) = \begin{cases} M_{N_1}(s_1) & \text{if } s_1 \in S_1 \\ M_{N_2}(s_2) & \text{if } s_2 \in S_2 \end{cases}
\end{array}$$

To describe the behaviour of a composed net, we will also have to compose sequences of actions in such a way that actions from some set A are synchronized. Let u and v be finite or infinite sequences over Σ and $A \subseteq \Sigma$. Then $u \parallel_A v$ is the set of all sequences w over Σ such that we can write u, v and w as sequences $u = u_1 u_2 \ldots$, $v = v_1 v_2 \ldots$ and $w = w_1 w_2 \ldots$ of equal finite or infinite length such that for all suitable $i = 1, 2, \ldots$ one of the following cases applies:

1. $u_i = v_i = w_i \in A$
2. $u_i = w_i \in (\Sigma - A)$ and $v_i = \lambda$
3. $v_i = w_i \in (\Sigma - A)$ and $u_i = \lambda$

In this definition, λ's are inserted into the decomposition of u and v to describe the interleaving of actions from $\Sigma - A$.

Parallel composition is maybe the most important operator for the modular construction of nets, but hiding and renaming are also essential. *Hiding* $A \subseteq \Sigma$ in N means changing all labels $a \in A$ to λ; it results in N/A; we write N/a instead of $N/\{a\}$ for a single $a \in \Sigma$. Similarly, w/A is obtained from a finite or infinite sequence w over Σ by removing all occurrences of actions from A. Clearly, $N/A/B = N/(A \cup B)$; we will freely combine several applications of hiding into one or split one into several.

Just as in [VK98], our relabelling is a bit more general than usual, which will be convenient for our examples; a *relabelling function* f maps actions from Σ to nonempty subsets of Σ and λ to $\{\lambda\}$. For a relabelling function f, let $dom(f) = \{a \in \Sigma \mid f(a) \neq \{a\}\}$, $cod(f) = \bigcup_{a \in dom(f)} f(a)$ and $\alpha(f) = dom(f) \cup cod(f)$.

The *relabelling* $N[f]$ of N with relabelling function f is obtained from N by replacing each transition t with $l(t) = a$ by as many copies as $f(a)$ has elements and labelling each copy by the respective element; the copies are connected to the places just as t.

For a relabelling f and $X \subseteq \Sigma$, we set $f(X) = \bigcup_{a \in X} f(a)$, and we set $f^{-1}(X) = \{a \in \Sigma \mid f(a) \cap X \neq \emptyset\}$. We can extend a relabelling homomorphically to finite or infinite sequences over Σ such that each sequence is mapped to a set of sequences.

Usually, a relabelling will map almost all $a \in \Sigma$ to $\{a\}$ – we say it is the identity for these a; then, we will only list the exceptions together with their respective images in the form $N[a_1 \to A_1, \ldots, a_n \to A_n]$. Again, we omit the braces of A_i, if it has only one element. Thus, $N[a \to b, b \to \{c, d\}]$ is the net N with each a changed to b and each b-labelled transition duplicated to a c- and a d-labelled copy; for this relabelling function f, $dom(f) = \{a, b\}$, $cod(f) = \{b, c, d\}$ and $\alpha(f) = \{a, b, c, d\}$; furthermore, $f(aceb) = \{bcec, bced\}$.

We let the unary operations of hiding and renaming bind stronger than parallel composition.

We now give some laws for our operations; basically the same were stated e.g. in [VK98], but for transition systems. These laws are based on isomorphism and should therefore hold whatever more detailed semantics one may choose; in particular, they are true for the fairness based preorder we will introduce in the next section.

Law 1 $(N_1 \parallel N_2) \parallel N_3 = N_1 \parallel (N_2 \parallel N_3)$

Law 2 $N_1 \parallel N_2 = N_2 \parallel N_1$

These laws will also be used freely without referencing them explicitly.

Law 3 $N[f][g] = N[f \circ g]$ where $(f \circ g)(a) = \bigcup_{b \in f(a)} g(b)$

Law 4 $N/A = N/(A \cup B)$ provided $\alpha(N) \cap B = \emptyset$

Law 5a $N[f]/A = N/A[f]$ provided $A \cap \alpha(f) = \emptyset$

Law 5b $N[a \to B]/A = N/(A \cup \{a\})$ provided $B \subseteq A$

Law 6 $(N_1 \parallel N_2)[f] = N_1 \parallel N_2[f]$ provided $\alpha(N_1) \cap \alpha(f) = \emptyset$

Law 7 $(N_1 \parallel N_2)/A = N_1 \parallel N_2/A$ provided $\alpha(N_1) \cap A = \emptyset$

Law 8 $(N_1 \parallel N_2)[f] = N_1[f] \parallel N_2[f]$ provided f only renames some actions to fresh actions, i.e. $f(a)$ is a singleton with $f(a) \cap (\alpha(N_1) \cup \alpha(N_2)) = \emptyset$ for all $a \in dom(f)$, and for different $a, b \in dom(f)$, $f(a) \neq f(b)$

We can now derive a law for α-conversion, i.e. the renaming of actions that are bound by hiding; applying Law 4 for $B = \{b\}$, Law 5b for $A = \{b\}$ and Law 8, we get:

Law 9 $(N_1 \parallel N_2)/a = (N_1[a \to b] \parallel N_2[a \to b])/b$ provided $b \notin \alpha(N_1) \cup \alpha(N_2)$

3 Fair Semantics

If a semantics is intended for specifying and checking liveness properties ('something good eventually happens'), one usually has to consider some sort of fairness. We will define a semantics that incorporates the progress assumption, also called weak fairness, i.e. the assumption that a continuously enabled activity should eventually occur. Continuous enabledness means that the respective 'processor' and all other resources are always available, and in a parallel system such an independent processor will certainly act eventually. Thus, our sort of fairness is met automatically, it does not have to be implemented; see [Fra86] for more on fairness. After the definition, we will determine a compositional semantics suitable for dealing with our sort of fairness.

In [Vog97], we defined a fair semantics and determined the coarsest compositional refinement of it for *safe* nets. This result does not directly carry over to general nets, but it can be generalized when we use a slightly peculiar definition of fairness; we will discuss this peculiarity in detail after the definition.

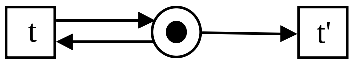

Figure 1

But first, we have to discuss the impact of read arcs on the progress assumption. Classically [Fra86], an infinite firing sequence $M_N[t_0\rangle M_1[t_1\rangle M_2\dots$ would be called fair if we have: if some transition t is enabled under all M_i for i greater than some j, then $t = t_i$ for some $i > j$. With this definition, the sequence t^ω of infinitely many t's would not be fair in the net of Figure 1, since t' is enabled under all states reached, but never occurs. But, in fact, t' is not continuously enabled, since every occurrence of t disables it momentarily, compare [Rei84, Vog95]; one could even say that the resource needed by t' is nearly always in use. Thus, t^ω should be fair. On the other hand, if t were on a read arc instead of a loop, t^ω should not be fair: t would only repeatedly check the presence of a resource without actually using it. To model this adequately, we will require in the definition of fairness that a continuously enabled t is enabled also *while* each t_i with $i > j$ is firing, i.e. enabled under $M_i - {}^\bullet t_i$.

Definition 3.1 For a transition t, a finite firing sequence $M_N[t_0\rangle M_1[t_1\rangle M_2\dots M_n$ is called *t-fair*, if M_n does not enable t. An infinite firing sequence $M_N[t_0\rangle M_1[t_1\rangle M_2\dots$ is called *t-fair*, if we have: For no j, t is enabled under all $M_i - {}^\bullet t_i$ for all $i > j$. If a finite or infinite firing sequence w violates the respective requirement, we say that t is *eventually enabled permanently* in w.

A finite or infinite firing sequence is *fair*, if it is t-fair for all transitions t of N; we denote the set of these sequences by $\mathit{FairFS}(N)$. The *fair language* of N is the set $\mathit{Fair}(N) = \{v \mid v = l(w) \text{ for some } w \in \mathit{FairFS}(N)\}$ of *fair traces.* □

What we require in the case of an infinite sequence is stricter than the more usual requirement that, if t is enabled under all $M_i - {}^\bullet t_i$ for i greater than some j, then $t = t_i$ for some $i > j$. For safe nets, these requirements coincide:

Proposition 3.2 *Let N be a net without self-concurrency (or, in particular a safe net), t a transition and $M_N[t_0\rangle M_1[t_1\rangle M_2 \ldots$ an infinite firing sequence. Assume further that if t is enabled under all $M_i - {}^\bullet t_i$ for i greater than some j, then $t = t_i$ for some $i > j$. Then this sequence is t-fair.*

Proof: Let $t = t_i$ for some i. By assumption on N, t is not enabled under $M_i - {}^\bullet t_i$; thus, for no j, t is enabled under all $M_i - {}^\bullet t_i$ for $i > j$. □

In our constructions, we have to work with nets that may not be safe; but in the end, we are only interested in safe nets. Thus, it is of no particular importance what a fair firing sequence of an unsafe net is; hence, we can choose a definition that is technically convenient. Technical convenience means in this case that we can obtain a fairness-respecting precongruence easily, i.e. in the same way as for safe nets.

It should be noted however that in the case of self-concurrency, the two variants of fairness differ, indeed. In the net of Figure 2, the firing sequence t^ω of infinitely many t's is not fair according to our definition, and in fact no firing sequence is (while t^ω would be fair with the usual requirement). As just stated, the main point is that this is irrelevant for the systems we want to study. Additionally, one could argue that firing sequences are not adequate to capture the progress assumption for general nets properly; instead, one could recommend step sequences and require that a fair step sequence of the net in Figure 2 had repeatedly steps consisting of two t's.

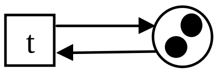

Figure 2

Next, we will determine the coarsest precongruence for parallel composition that respects fair-language-inclusion; this is just the right relation if we want to build systems compositionally and are interested in the fair language. Theorems 3.4 and 3.5 were to my knowledge first obtained by Robert Gold [Gol88], surveyed in [Vog92, page 69] and improved in several ways in [Vog97]. Here, we generalize them from safe nets to general nets (with read arcs). Recall, that we use a slightly different form of parallel composition here, but that this is merely a convenience. There is one consequence, though: in the definition of a fair implementation below, we additionally require the nets under consideration to have the same alphabets. This requirement is very natural in our setting where the alphabet of a net determines those actions that are synchronized in a parallel composition.

Definition 3.3 A net N_1 is a *fair implementation* of a net N_2, if $\alpha(N_1) = \alpha(N_2)$ and $\mathit{Fair}(N_1 \| N) \subseteq \mathit{Fair}(N_2 \| N)$ for all nets N.

For a net N, the *fair failure* semantics is the set of the *fair refusal pairs* defined by $\mathcal{FF}(N) = \{(v, X) \mid X \subseteq \Sigma$ and $v = l(w)$ for some, possibly infinite, firing sequence w that is t-fair for all transitions t with $l(t) \in X \cup \{\lambda\}\}$.

We write $N_1 \leq_{\mathcal{FF}} N_2$, if N_1 and N_2 have the same alphabet and $\mathcal{FF}(N_1) \subseteq \mathcal{FF}(N_2)$. If $N_1 \leq_{\mathcal{FF}} N_2$ and $N_2 \leq_{\mathcal{FF}} N_1$, we write $N_1 =_{\mathcal{FF}} N_2$ and call the nets *fair-congruent.* □

The motivation for this definition is as follows: assume N_1 is a fair implementation of the specification N_2, N_2 is a component of a parallel system and we replace this component by N_1; then we will get only fair behaviour that is allowed by N_2, i.e. that is possible when N_2 is used.

The intuition for $(v, X) \in \mathcal{FF}(N)$ is that all actions in X can be *refused* when v is performed – in the sense that fairness does not force performance of these actions; yet in other words, these actions are treated correctly w.r.t. fairness. This is essentially (up to divergence, i.e. infinite internal runs) the same intuition as for ordinary failure semantics [BHR84]: after a finite run, fairness forces the performance of some enabled action while disabled ones can be refused.

Theorem 3.4 *For nets N_1 and N_2 with $\alpha(N_1) \cap \alpha(N_2) = A$ we have*

$$\begin{aligned}\mathcal{FF}(N_1 \| N_2) \quad = \quad & \{(v, X) \mid \exists (v_i, X_i) \in \mathcal{FF}(N_i),\ i = 1, 2 : \\ & \quad v \in v_1 \|_A v_2 \ \textit{and}\ X \subseteq ((X_1 \cup X_2) \cap A) \cup (X_1 \cap X_2)\}\end{aligned}$$

Proof: Similar proofs can be found e.g. in the full version of [Vog97], and the proof is simpler than the proof of the corresponding Theorem 5.10 there. The essential observations are the following (where $^\bullet * = \emptyset$ and $*$ is always enabled, but does not change the marking if 'fired'): the reachable markings of $N_1 \| N_2$ can be written as $M_1 \dot\cup M_2$ with M_i a reachable marking of N_i - though possibly not all combinations really turn up; $M_1 \dot\cup M_2 [(t_1, t_2)\rangle$ in $N_1 \| N_2$ if and only if $M_1[t_1\rangle$ in N_1 and $M_2[t_2\rangle$ in N_2 – and in this case $M_1 \dot\cup M_2 [(t_1, t_2)\rangle M_1' \dot\cup M_2'$ in $N_1 \| N_2$ for the markings M_1' and M_2' with $M_1[t_1\rangle M_1'$ in N_1 and $M_2[t_2\rangle M_2'$ in N_2; if $M_1 \dot\cup M_2 [(t_1, t_2)\rangle$ in $N_1 \| N_2$, then $(M_1 \dot\cup M_2) - {}^\bullet(t_1, t_2)$ enables (t_1', t_2') if and only if $(M_1 - {}^\bullet t_1)[t_1'\rangle$ and $(M_2 - {}^\bullet t_2)[t_2'\rangle$.

The crucial point, where the subtlety of our unusual fairness definition comes into play, concerns inclusion: assume we have constructed (v, X) from (v_1, X_1) and (v_2, X_2) as described on the right hand side of the above equation. Using the above observations, we can construct from suitable firing sequences w_1 and w_2 underlying v_1 and v_2 a firing sequence $w = (t_{11}, t_{21})(t_{12}, t_{22})(t_{13}, t_{23}) \dots$ of $N_1 \| N_2$ that projects to w_1 and w_2, i.e. it satisfies the following (where $*$ in a sequence is treated as λ and thus simply deleted): $l(w) = v$, $w_1 = t_{11} t_{12} t_{13} \dots$ and $w_2 = t_{21} t_{22} t_{23} \dots$. Assume further that $a \in X_1 \cap A$, and assume by way of contradiction that w is not (t_1', t_2')-fair for some a-labelled transition (t_1', t_2') of $N_1 \| N_2$.

Hence, for some j, (t_1', t_2') is enabled under all $(M_{1i} \dot\cup M_{2i}) - {}^\bullet(t_{1i}, t_{2i})$ with $i > j$ and suitable M_{1i} and M_{2i}. By the above, this implies that t_1' is enabled under all $M_{1i} - {}^\bullet t_{1i}$ with $i > j$, a contradiction.

With the more usual definition of fairness, this would not be a contradiction. Instead, it would imply that t_1' is one of these t_{1i}; but this does not imply that the respective t_{2i} is t_2', and the proof fails. $\square$

Remark: An alternative approach to get a similar result could be to regard firing sequences that violate safeness as catastrophic, and hence to ignore what happens after such a violation; this could lead to a semantics with two sets for the two types of firing sequences that do or do not violate safeness; compare [Vog92, Section 3.3] for a similar approach that incorporates safeness into a semantics.

This approach would have some similarity with the treatment of *divergence*, i.e. infinite internal firing sequences, in ordinary failure semantics. See [Vog97] for a discussion how $\mathcal{FF}$-semantics treats divergence. □

The last theorem is an important step for the compositionality result we have been aiming for.

Theorem 3.5 *i) For nets N_1 and N_2, N_1 is a fair implementation of N_2 if and only if $N_1 \leq_{\mathcal{FF}} N_2$.*

ii) For nets with some fixed alphabet, inclusion of $\mathcal{FF}$-semantics is fully abstract w.r.t. fair-language inclusion and parallel composition of nets, i.e. it is the coarsest precongruence for parallel composition that refines fair-language inclusion.

Proof: i) The if-part is clear from Theorem 3.4 and because $v \in \mathit{Fair}(N)$ iff $(v, \Sigma) \in \mathcal{FF}(N)$. For the only-if part, let $A = \alpha(N_1) = \alpha(N_2)$, $x \in \Sigma - A$ and $(v, X) \in \mathcal{FF}(N_1)$, where we may assume $X \subseteq A$, since actions not in A are treated fairly in both nets in any case. We construct an environment net N with the following parts for each $a, b, c, d \in A$, see Figure 3. If a occurs $m \in I\!\!N$ times in v, N contains a chain

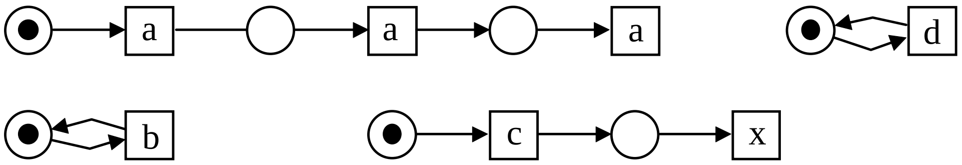

Figure 3

of m a-transitions (as shown in Figure 3 for $m = 3$); if b occurs infinitely often in v, N contains a b-transition on a marked loop; for $c \in X$, N contains a cx-chain (see Figure 3). Furthermore, for each d not occurring in v or X, N contains a d-transition on an unmarked loop. Observe that $\alpha(N) = A \cup \{x\}$, hence in a parallel composition with N_1 or N_2 all actions from A will be synchronized, and thus will be refused in a run if one of the components refuses them. Obviously, N can perform v without using the cx-chains such that all $e \in \Sigma - X$ are treated fairly, i.e. $(v, \Sigma - X) \in \mathcal{FF}(N)$. Hence, $(v, \Sigma) \in \mathcal{FF}(N \| N_1)$ by 3.4 and $v \in \mathit{Fair}(N \| N_1) \subseteq \mathit{Fair}(N \| N_2)$, thus $(v, \Sigma) \in \mathcal{FF}(N \| N_2)$. Applying 3.4 again, there must exist a suitable $(v, Y) \in \mathcal{FF}(N)$, where in particular we must have $x \in Y$ since $x \notin A$. Thus, N must perform v without using any of the c-transitions from the cx-chains; the maximal set Y for this case is $\Sigma - X$, i.e. we must have $(v, X) \in \mathcal{FF}(N_2)$.

ii) By definition of fair implementation, a precongruence for parallel composition that refines fair-language inclusion must also refine the fair-implementation relation. On the other hand, the latter is a precongruence by Part i) and 3.4 and it refines fair-language inclusion since $v \in \mathit{Fair}(N)$ if and only if $(v, \Sigma) \in \mathcal{FF}(N)$. Thus, the latter is the coarsest such precongruence. □

It is not difficult to show the following result concerning precongruence for relabelling and hiding.

Theorem 3.6 *The relation $\leq_{\mathcal{FF}}$ is a precongruence w.r.t. relabelling and hiding. More precisely, for a net N we have:*

- $\mathcal{FF}(N[f]) = \{(f(w), X) \mid (w, f^{-1}(X)) \in \mathcal{FF}(N),\ \textit{where}\ w \in \Sigma^\infty, X \subseteq \Sigma\}$
- $\mathcal{FF}(N/A) = \{(w/A, X) \mid (w, X \cup A) \in \mathcal{FF}(N),\ \textit{where}\ w \in \Sigma^\infty, X \subseteq \Sigma\}$

In [Vog97], it is shown that for safe nets $\leq_{\mathcal{FF}}$ is decidable. Further, an operation is considered that will be of interest later. We define this operation, state that it preserves safety and quote a result from [Vog97].

Definition 3.7 An *elongation* of N is obtained by choosing a transition t, adding a new unmarked place s and a new λ-labelled transition t' with ${}^\bullet t' = \{s\}$ and $t'^\bullet = t^\bullet$ and, finally, redefining $t^\bullet$ by $t^\bullet := \{s\}$. □

Theorem 3.8 *If a net N_2 is an elongation of a net N_1, then one of the nets is safe if and only if the other one is; in this case, $N_1 =_{\mathcal{FF}} N_2$.*

Remark: The second statement of this theorem may fail for unsafe nets. Consider an a-labelled transition on a loop with a place carrying two tokens. This net has no fair traces, but after elongation one can fire repeatedly the a-labelled transition twice and the internal transition twice, giving the fair trace a^ω. □

We close this section with a notion, also taken from [Vog97] (and similar to one used for ordinary failure semantics), that will make Definition 5.1 more suggestive. $(v, X) \in \mathcal{FF}(N)$ means that N can perform v in such a way that all internal actions and all actions in X are treated fairly. Hence, $(v, X) \notin \mathcal{FF}(N)$ means that either N cannot perform v in such a way that all internal actions are treated fairly or it can, but whichever way it performs v, it treats some action in X unfairly. The latter means that some $x \in X$ is continuously enabled from some point onward; if N is on its own, it certainly performs such an x – but as a component of a larger system, N simply offers such an x. We therefore define:

Definition 3.9 If for a net N and some $(v, X) \in \Sigma^\infty \times \mathcal{P}(\Sigma)$ we have $(v, X) \notin \mathcal{FF}(N)$, then we say that N **surely offers** (some action of) X **along** v. □

If N **surely offers** X **along** v and, in a run of a composed system, N as a component performs v while the environment offers in this run each action in X, then some action in X will be performed in this run.

4 Partial S-Invariants

Corresponding to the interests of this paper, we will only consider a restricted form of S-invariants (and consequently of partial S-invariants) defined as follows.

Definition 4.1 Let N be a net; a set P of places has *value* n (under a marking M) if the places in P carry together n tokens under the initial marking (under M resp.). An *S-invariant* of N is a set $P \subseteq S$ of value 1 such that for every transition t we have $|P \cap {}^\bullet t| = |P \cap t^\bullet|$.

A *partial S-invariant* of N with input $I \subseteq \alpha(N)$ and output $O \subseteq \alpha(N)$ is a set P such that for every transition t we have: if the label of t is in I, then $|P \cap t^\bullet| - |P \cap {}^\bullet t| = 1$; if the label of t is in O, then $|P \cap {}^\bullet t| - |P \cap t^\bullet| = 1$; if the label of t is neither in I nor in O, then $|P \cap {}^\bullet t| = |P \cap t^\bullet|$. We call such a P an (I, O, n)-invariant, if additionally P has value n.

N is *covered* by S-invariants if each place is contained in an S-invariant. N is *covered* by partial S-invariants $P_1, P_2, \ldots, P_n$, if either $n = 0$ and N is covered by S-invariants or $n > 0$ and each place is contained in an S-invariant or some P_i.

For $I \subseteq \Sigma$, $O \subseteq \Sigma$ and $n \in I\!N_0$, the (I, O, n)*-component* $C(I, O, n)$ is a net consisting of a place s with n tokens, an i-labelled transition t with $t^\bullet = \{s\}$ and ${}^\bullet t \cup \hat{t} = \emptyset$ for each $i \in I$ and an o-labelled transition t with ${}^\bullet t \cup \hat{t} = \{s\}$ and $t^\bullet = \emptyset$ for each $o \in O$. □

We only use S-invariants where we just count the places in the pre- and postset of a transition. In general S-invariants, each place has a weight and, instead of counting as in $|P \cap {}^\bullet t|$, one adds up the respective weights; thus, in *our* S-invariants this weight is always one. Furthermore, the value does not have to be 1 in general S-invariants. Similarly, we just count places in the definition of partial S-invariants, and we restrict attention to the case where firing of a transition changes the value by 1, 0 or -1. There is no problem in generalizing the definition to weighted places and general changes, except that the notation would get heavier and this generality is not needed here. Also, one would need arc weights to define the analogue of (I, O, n)-components.

The following result is well-known and easy:

Theorem 4.2 *If P is an S-invariant of a net N, then P has value 1 under all reachable markings. If N is covered by S-invariants, then N is safe.*

Next, we state a number of properties regarding our new notions; their proofs are easy. (In particular, 4 follows from 3.)

Proposition 4.3

1. *If P_i is an (I_i, O_i, n_i)-invariant of N_i, $i = 1, 2$, such that $I_1 \cap I_2 = \emptyset = O_1 \cap O_2$, then $P_1 \cup P_2$ is an $((I_1 \cup I_2) - (O_1 \cup O_2), (O_1 \cup O_2) - (I_1 \cup I_2), n_1 + n_2)$-invariant of $N_1 \| N_2$. In particular, P_1 or P_2 can be $\emptyset$, which is an $(\emptyset, \emptyset, 0)$-invariant.*

2. *If N_i is covered by partial S-invariants $P_{i1}, P_{i2}, \ldots, P_{in}$, $i = 1, 2$, then $N_1 \| N_2$ is covered by the partial S-invariants $P_{11} \cup P_{21}, P_{12} \cup P_{22}, \ldots, P_{1n} \cup P_{2n}$.*

3. *An $(\emptyset, \emptyset, 1)$-invariant is an S-invariant.*

4. *If N is covered by partial S-invariants $P_1, P_2, \ldots, P_n$, $n > 0$ and P_1 is an $(\emptyset, \emptyset, 1)$-invariant, then N is covered by partial S-invariants $P_2, \ldots, P_n$.*

5. *$C(I, O, n)$ is covered by an (I, O, n)-invariant.*

Corollary 4.4 *If N is covered by an (I, O, n)-invariant and $m \in I\!N_0$ with $m+n=1$, then $N \| C(O, I, m)$ is safe.*

Proof: By Parts 5 and 2 of 4.3, $N \| C(O, I, m)$ is covered by a partial S-invariant that according to Part 1 is an $(\emptyset, \emptyset, 1)$-invariant. By Part 4 $N \| C(O, I, m)$ is covered by S-invariants and, thus, safe by 4.2. □

Partial S-invariants and the notion of covering are also consistent with relabelling and hiding in the following sense.

Proposition 4.5 *1. Let N be a net, $a \in \Sigma$ and $A \subseteq \Sigma$, such that $\alpha(N) \cap A \subseteq \{a\}$, and let f be the relabelling that maps a to A and is the identity otherwise.*

If P is an (I, O, n)-invariant of N, then P is an $(f(I), f(O), n)$-invariant of $N[f]$. If N is covered by partial S-invariants $P_1, P_2, \ldots, P_n$, then $N[f]$ is covered by the same partial S-invariants.

2. Let P be an (I, O, n)-invariant of N that does not meet *$A \subseteq \Sigma$, i.e. $I \cap A = O \cap A = \emptyset$. Then P is an (I, O, n)-invariant of N/A. If N is covered by partial S-invariants $P_1, P_2, \ldots, P_n$ that do not meet A, then N/A is covered by the same partial S-invariants.*

We close this section with the key result that will allow to insert some sort of interface description into a parallel composition. If N has an (I, O, n)-invariant, then $C(I, O, n)$ is a sort of abstraction of N; hence, adding it to N as parallel component does not change the behaviour as we show now.

Proposition 4.6 *If N has an (I, O, n)-invariant P, then $N \| C(I, O, n) =_{\mathcal{FF}} N$.*

Proof: Let s be the place of $C(I, O, n)$; we relate each marking M of N to the marking M' of $N \| C(I, O, n)$ that additionally marks s with n tokens where n is the value of P under M.

If M and M' are related, then M enables t if and only if M' enables t (identifying the pairs (t_1, t_2) of the parallel composition with t_1). The if-part of this is obvious, since $N \| C(I, O, n)$ just has an additional place; the only-if-part is obvious unless t has its label in O, in which case it removes a token from P, i.e. s has at least one token as required. From the definition, it is clear that firing t gives markings that are related again.

Furthermore, if M and M' are related enabling some t, then $M - {}^\bullet t'$ enables t if and only if $M' - {}^\bullet t'$ enables t. The proof is essentially as above observing that, if t' removes a token from s, then it removes at least one token from P. (Hence, $M' - {}^\bullet t'$ has at least as many tokens on s as the marking related to $M - {}^\bullet t'$.)

From these statements, we see that the two nets have the same firing sequences and that a transition t is eventually enabled permanently in w in one net if and only if it is in the other. This implies the result. □

This proposition can be applied in a situation where we want to reduce some $N \| N'$ to a smaller fair-congruent net. It might even be that – while N' is quite manageable, in particular not too large – the precise definition of N and its size depend on a parameter, i.e. its size may be arbitrarily large. If we know at least that N has an (I, O, n)-invariant, then $N \| N' =_{\mathcal{FF}} N \| C(I, O, n) \| N'$ by 4.5 and 3.5, i.e. we can insert $C(I, O, n)$ into the parallel composition; now we might be able to reduce $C(I, O, n) \| N'$ in a way that would not be possible for N' in isolation; if the component $C(I, O, n)$ perseveres (like a catalyst), we can remove it after the reduction. The following corollary describes this compositional reduction, also for the more general case of using several partial S-invariants. Observe that 4.6 is also valid for language-equivalence or bisimilarity in place of fair-congruence; hence, partial S-invariants also support the reduction method if these congruences are used.

In this method, $C(I, O, n)$ is an interface description; as an abstraction of N, $C(I, O, n)$ restricts the behaviour of N' to what is relevant in $N \| N'$. Important is that this interface description is verified on N syntactically, while the reduction deals with the behaviour of N' only, and not with that of N. This is in contrast to [GSL96], where some interface description is guessed, used in the reduction of N' to N'' and then verified during the further reduction of $N \| N''$. The latter considers the behaviour of N, which is not possible in the fixed-point approach where N is parametric, i.e. not completely known.

Corollary 4.7 *1. Assume that N has an (I, O, n)-invariant and $C(I, O, n) \| N'$ $=_{\mathcal{FF}} C(I, O, n) \| N''$ or $C(I, O, n) \| N' =_{\mathcal{FF}} N''$. Then $N \| N' =_{\mathcal{FF}} N \| N''$.*

2. Assume that N has a family of partial S-invariants, C is the parallel composition of the respective (I, O, n)-components and C' is the parallel composition of a subfamily of these (I, O, n)-components. If $C \| N' =_{\mathcal{FF}} C' \| N''$, then $N \| N' =_{\mathcal{FF}}$ $N \| N''$.

5 Liveness Properties of MUTEX-Solutions

This section repeats necessary material from [BV98] regarding the correctness of MUTEX-solutions and also introduces our first application example. First, we will formulate a correctness specification based on the $\mathcal{FF}$-semantics consisting of a safety and a liveness requirement. *Safety* requires that no two users are in their critical sections at the same time; if one user enters, then he must leave before another enter is possible. Our definition of *liveness* is explained after Definition 5.1; we only remark at this point that a MUTEX-solution can only guarantee that each requesting user can enter his critical section, if in turn each user e.g. guarantees to leave after entering.

Definition 5.1 We call a finite or infinite sequence over $I_n = \{r_i, e_i, l_i \mid i = 1, \ldots, n\}$ *legal* if r_i, e_i and l_i only occur cyclically in this order for each i. An *n-MUTEX net* is a net N with $l(T) \subseteq \{r_i, e_i, l_i \mid i = 1, \ldots, n\} \cup \{\lambda\}$.

Such a net is a *correct n-MUTEX-solution*, if N satisfies *MUTEX-safety*, i.e. e- and l-transitions only occur alternatingly in a legal trace, and satisfies *MUTEX-liveness* in the following sense. Let $w \in I_n^* \cup I_n^\omega$ be legal and $1 \leq i \leq n$; then:
1. Each e_i in w is followed by an l_i, or N **surely offers** $\{l_i\}$ **along** w.

2. Assume each e_j is followed by l_j in w. Then either each r_i is followed by e_i or N **surely offers** X **along** w where X consists of those e_j where some r_j in w is not followed by e_j.
3. Assume each r_j is followed by e_j and each e_j is followed by l_j in w. Then either r_i occurs and each l_i is followed by another r_i in w or N **surely offers** $\{r_i\}$ **along** w.
□

An n-MUTEX net N is used in a complete system consisting of N and its environment comprising the users, and these two components synchronize over I_n. The first part of MUTEX-liveness says that, if user i enters (performs e_i together with the scheduler N), later tries to leave (enables an l_i-transition) and does not withdraw (does not disable the transition again), then he will indeed leave; otherwise l_i would be enabled continuously in the complete system violating fairness. (Technically, recall how the refusal sets of fair refusal pairs are composed according to Theorem 3.4: the complete system is fair, i.e. Σ is refused, only if one of the components refuses l_i.)

In other words, if user i does not leave again, then he is not willing to leave since l_i is offered to him. This is a user misbehaviour, but the behaviour of the scheduler N is correct. As a consequence, if N satisfies Part 1, we can assume that each e_j is followed by l_j. Under this assumption, the second part of MUTEX-liveness says that each request of i is satisfied, unless some requesting user is permanently offered to enter. In the latter case, that user is misbehaving by not accepting this offer, and again N is working correctly.

Now we can assume that each request is satisfied. Under this assumption, i requests infinitely often or N at least offers him to request. The latter is not a user misbehaviour because each user should be free to decide whether he wants to request.

The following is obvious from the definitions.

Proposition 5.2 *If $N_1 \leq_{\mathcal{FF}} N_2$ and N_2 satisfies MUTEX-safety, MUTEX-liveness resp., then also N_1 satisfies MUTEX-safety, MUTEX-liveness resp.*

For token-passing solutions, MUTEX-safety is usually easy to prove with an S-invariant; for the two families of solutions we will treat, sufficient arguments for MUTEX-safety are given in [BV98]. Hence, we will concentrate on proving MUTEX-liveness, which is more difficult. As already mentioned, application of the fixed-point approach does not seem feasible, since the more users an n-MUTEX net has to serve, the more visible actions it has.

The surprising fact is that, under some symmetry assumptions, it is enough to check a version where only the actions of one user are visible. It should be pointed out that the respective Theorem 5.5 is somewhat fragile: it was necessary to modify the very similar correctness definition of [Vog97] to make it work. For our two families of solutions, [BV98] also gives sufficient arguments that each of their nets are user-symmetric in the following sense.

Definition 5.3 A *quasi-automorphism* ϕ of a net N is an isomorphism of the net graph of N onto itself that maps the initial marking to a reachable marking, i.e.: ϕ is a bijection of $S \cup T$ onto itself such that $\phi(S) = S$, $\phi(T) = T$, $(x, y) \in F \Leftrightarrow$

$(\phi(x), \phi(y)) \in F$, $(x, y) \in R \Leftrightarrow (\phi(x), \phi(y)) \in R$, and $\phi(M_N)$ is a reachable marking. (Here, M_N is regarded as a set.) Note that ϕ ignores the labelling.

A *user symmetry* (ϕ, π) of an n-MUTEX net N consists of a quasi-automorphism ϕ of N and a permutation π of $\{1, \ldots, n\}$ such that, first, $\phi(M_N)$ is reachable with a legal trace where for each i the last i-indexed action (if any) is l_i and, second, for all $i = 1, \ldots, n$ and all $t \in T$ we have $l(t) = r_i \Leftrightarrow l(\phi(t)) = r_{\pi(i)}$, $l(t) = e_i \Leftrightarrow l(\phi(t)) = e_{\pi(i)}$, and $l(t) = l_i \Leftrightarrow l(\phi(t)) = l_{\pi(i)}$.

An n-MUTEX net is *user-symmetric* if, for all $i, j \in \{1, \ldots, n\}$, it has a user symmetry (ϕ, π) with $\pi(i) = j$. □

Our first family of solutions is attributed to Le Lann; each of its nets is a ring of components, one for each user. Figure 4(a) shows the component LLU of the first user (except that the actions r, e and l should be indexed with 1, but we identify r with r_1 etc.). This user owns the access-token, the token on the right, while the other users look the same, except that they do not have this token. The first user can $r$equest access with r (i.e. by firing the r-labelled transition t_r) and $e$nter the critical section with e. When he $l$eaves it with l, he passes the token to the next user, i.e. the l-transition must be merged with the p-transition of the next user; p stands for previous, the respective transition produces the access-token coming from the previous user.

If the first user is not interested in entering the critical section, the token is passed by the n-transition to the $n$ext user; i.e. the n-transition must also be merged with the p-transition of the next user and then hidden. It is important that the n-transition checks the token on ${}^\bullet t_r$ with a read arc, since this way the user is not prevented from requesting in a firing sequence with infinitely many checks. Intuitively, Le Lann's solution is correct, since the access-token is passed around as just explained, and if a user has requested and the token reaches the respective ring component, the user will enter and leave his critical section before passing the token on.

For a Le-Lann-ring built as just explained, it should be clear that after firing the n-transition we get a symmetric marking where the second user owns the access-token.

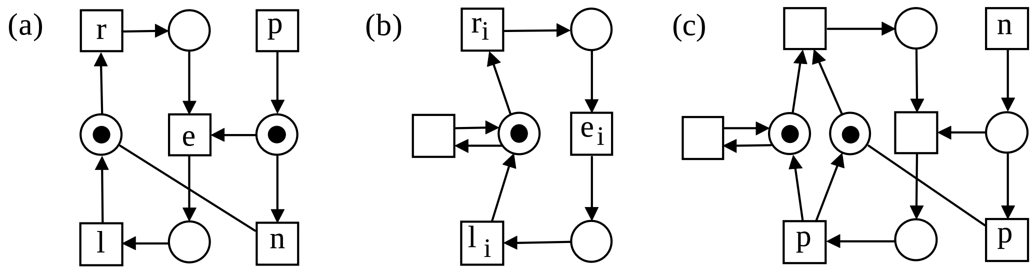

Figure 4

We next define the first-user view of a solution. For this we assume that each user except the first one has a standard behaviour modelled by the i-th *standard user* SU_i shown in Figure 4(b): when non-critical, such a user works internally or requests; after requesting, he enters and leaves as soon as possible. Then we abstract away all visible behaviour of these users with a suitable hiding.

Note that only performing this hiding would have a very similar effect; but the user then would also always request as soon as possible, i.e. he would certainly request under the progress assumption, which clearly is not the standard behaviour we want to deal with. The first-user view will allow to apply the fixed-point approach.

Definition 5.4 The *first-user view* $FUV(N)$ of a net N is $(N\|(SU_2\|\ldots\|SU_n))/\{r_i, e_i, l_i \mid i = 2, \ldots, n\}$. □

The first-user view of Le Lann's ring is a ring where the first component looks like Figure 4(a), while all other components look like LSU shown in (c), except for the labelling; also, we have omitted the two unmarked places of SU_i that are simply duplicates in LSU. The labelling of LSU has been chosen such that we can directly construct the first-user view of each Le-Lann-ring from LLU and a suitable number of copies of LSU. Only these nets we have to study due to the following theorem.

Theorem 5.5 *Assume that a safe n-MUTEX net N is user-symmetric and satisfies MUTEX-safety. Then, N is a correct n-MUTEX solution if $FUV(N)$ is a correct 1-MUTEX solution.*

6 Correctness Proofs

In this section, we will develop the fixed-point approach for two families of solutions to the MUTEX-problem already discussed in [BV98]. In both, an access-token is passed around which guarantees mutual exclusion.

6.1 Le Lann's Ring

We first describe formally how the first-user view of a Le-Lann-ring as explained in the previous section can be constructed from LLU and several copies of LSU, which are shown in Figure 4. We first define the 'Le-Lann-chain' LLC_n inductively, which is a chain of n copies of LSU.

$$LLC_1 = LSU$$

$$LLC_{n+1} = (LSU[n \to p']\|LLC_n[p \to p'])/p'$$

This chain has one n-labelled transition, which moves the access-token to the chain, and two p-labelled transitions 'at the other end', which remove the access-token from the chain. Clearly, this net is not safe, since the n-labelled transition can fire several times in a row. Now we close these chains to rings with LLU:

$$LL_n = (LLU\|LLC_{n-1}[n \to \{l, n\}])/\{p, n\}$$

We first observe:

Proposition 6.1 *1. LSU is covered by an $(\{n\}, \{p\}, 0)$-invariant. LLU is covered by a $(\{p\}, \{l, n\}, 1)$-invariant.*

2. LLC_n is covered by an $(\{n\}, \{p\}, 0)$-invariant and its alphabet is $\{n, p\}$ for all n.

3. LL_n is safe for all n.

Proof: 1. There are two circuits in LSU, each containing one of the marked places; they are S-invariants. The places on the two paths from the n-labelled to the two p-labelled transitions form an $(\{n\},\{p\},0)$-invariant. The case of LLU is very similar.

2. The claim about the alphabet follows by an easy induction. The first claim can also be shown by induction, where $i = 1$ follows from 1. By 1. and 4.5.1 and induction, $LSU[n \to p']$ is covered by a $(\{p'\},\{p\},0)$-invariant and $LLC_n[p \to p']$ is covered by an $(\{n\},\{p'\},0)$-invariant. Thus, by 4.3.1 and .2 and 4.5.2, LLC_{n+1} is covered by an $(\{n\},\{p\},0)$-invariant.

3. By 1., LLU is covered by a $(\{p\},\{l,n\},1)$-invariant. By 2. and 4.5.1 (observe that $l \notin \alpha(LLC_{n-1})$), $LLC_{n-1}[n \to \{l,n\}]$ is covered by an $(\{l,n\},\{p\},0)$-invariant. Now by 4.3.1 and .2, $LLU \| LLC_{n-1}[n \to \{l,n\}]$ is covered by an $(\emptyset,\emptyset,1)$, and by 4.5.2, 4.3.4 and 4.2, LL_n is safe. □

From the observations in [BV98] – underpinned by the safeness just shown – we can apply Theorem 5.5 to each Le-Lann-ring assuming that MUTEX-safety is satisfied; it is planned to refine the notion of partial S-invariant such that it supports the proof of MUTEX-safety. Hence, it remains to prove that LL_n satisfies MUTEX-liveness, where we identify r, e and l with r_1, e_1 and l_1; this has already been shown for $n = 2, 3, 4$ in [BV98]. We start with two lemmata.

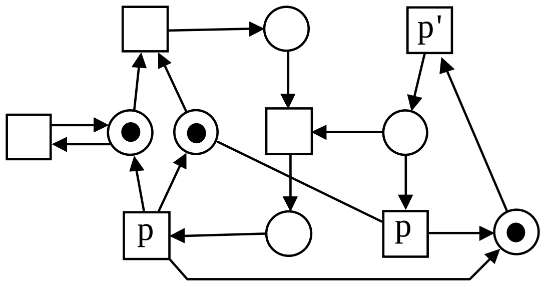

Figure 5

Lemma 6.2 $LSU[n \to p'] \| C(\{p\},\{p'\},1) =_{\mathcal{FF}} C(\{p'\},\{p\},0) \| C(\{p\},\{p'\},1)$

Proof: The left-hand-side net N_1 is shown in Figure 5, the right-hand-side net N_2 is simply a circuit with a p'- followed by a p-transition. We argue that the $\mathcal{FF}$-semantics of both nets is $\{((p'p)^\omega, X) \mid X \subseteq \Sigma\} \cup \{(w,X) \mid w \in (p'p)^* \wedge p' \notin X \subseteq \Sigma\} \cup \{(w,X) \mid w \in (p'p)^* p' \wedge p \notin X \subseteq \Sigma\}$.

It is clear that the visible behaviour of both nets is an alternating sequence of p' and p. If the sequence is infinite, all transitions are treated fairly (provided we repeatedly fire the loop-transition of N_1), hence arbitrary sets can be refused. If the sequence is empty or ends with p, all transitions except the p'-labelled one are treated fairly (with the same provision for N_1), hence arbitrary sets not containing p' can be refused.

That the analogous statement holds for sequences ending with p' is clear for N_2. For N_1, the p'-transition is treated fairly; now either the loop-transition fires infinitely often and the right p-transition is continuously enabled or the other two internal transitions must be fired, since we consider only firing sequences that are fair for all internal transitions; in the latter case the left p-transition is continuously enabled, and we are done. $\square$

The next lemma shows how to reduce one LSU-component. This only works due to the presence of C, which will arise from a partial S-invariant in the proof of Theorem 6.4; C also ensures safeness of the nets the lemma deals with; see the discussion at the end of this subsection.

Lemma 6.3 *Let* $C = C(\{l,n\},\{p'\},0)$*; then* $(LLU\|LSU[n \to p'])/p \parallel C =_{\mathcal{FF}} LLU[p \to p']\|C$.

Proof: By Law 7, $(LLU\|LSU[n \to p'])/p \parallel C =_{\mathcal{FF}} (LLU\|LSU[n \to p']\|C)/p$. Now LLU has a $(\{p\},\{l,n\},1)$-invariant by 6.1.1, C has a $(\{l,n\},\{p'\},0)$-invariant by 4.3.5 and, hence, their parallel composition has a $(\{p\},\{p'\},1)$-invariant by 4.3.1. Thus, by applying Corollary 4.7 and Lemma 6.2, we can reduce $LSU[n \to p']$ to $C(\{p'\},\{p\},0)$ and arrive at $(LLU\|C(\{p'\},\{p\},0)\|C)/p$, which by Law 7 again is fair-congruent to $(LLU\|C(\{p'\},\{p\},0))/p \parallel C$. In the latter net, the unique p'-labelled transition has as postset the place of $C(\{p'\},\{p\},0)$, which in turn has an internal transition as postset; considering this, one sees that the net is simply an elongation of the right-hand-side net in the lemma, and we are done by Theorem 3.8. $\square$

Now we can apply the central Corollary 4.7 again to obtain the fixed-point result and the correctness we are aiming for in this subsection.

Theorem 6.4 *For* $n > 2$, $LL_n =_{\mathcal{FF}} LL_{n-1}$. *For* $n > 1$, *Le Lann's ring is a correct MUTEX-solution.*

Proof: Once we have shown the congruence, we have that each LL_n is fair-congruent to LL_2, which has been shown to satisfy MUTEX-liveness in [BV98]. Hence, each LL_n satisfies MUTEX-liveness by 5.2, and the second part follows with Theorem 5.5 as explained above.

Thus, we will transform

$$LL_n = (LLU \parallel (LSU[n \to p']\|LLC_{n-2}[p \to p'])\,/p'[n \to \{l,n\}])\,/\{p,n\}$$

preserving fair-congruence. First, we can commute the hiding of p' with the following renaming by Law 5a and move it out of the outer brackets by Law 7 (since $p' \notin \alpha(LLU)$) obtaining

$$(LLU \parallel (LSU[n \to p']\|LLC_{n-2}[p \to p'])\,[n \to \{l,n\}])\,/\{p',p,n\}.$$

Since $l,n \notin \alpha(LSU[n \to p'])$, we can move the right-most renaming to the left by Law 6; since the resulting component $L = LLC_{n-2}[p \to p'][n \to \{l,n\}]$ does not have p in its alphabet due to renaming, we can apply Law 7 to move $/p$ and get

$$((LLU\|LSU[n \to p']/p) \parallel LLC_{n-2}[p \to p'][n \to \{l,n\}])\,/\{p',n\}.$$

The component L has an $(\{l,n\},\{p'\},0)$-invariant by propositions 6.1.2 and 4.5.1. With Lemma 6.3 and Corollary 4.7, we obtain

$$(LLU[p \to p'] \parallel LLC_{n-2}[p \to p'][n \to \{l,n\}])\ /\{p',n\}.$$

By Law 3, we can commute the second renaming with the third one and then suppress it by Law 9; with the definition we arrive at LL_{n-1}. □

The essential step in this proof is to reduce one of the LSU-components. To prove this reduction directly would mean to show $(LLU \| LSU[n \to p'])/p =_{\mathcal{FF}} LLU[p \to p']$, but we have used partial S-invariants and Lemma 6.3 – because this congruence is actually wrong: only the left-hand side has the fair refusal pair $(p'(p'n)^\omega, \{n\})$. To see this, observe that, in $LLU[p \to p']$, n is permanently enabled after the first p' and thus cannot be refused. In the other net, we can have a token on the rightmost place of LSU permanently after the first p'; this gives a behaviour that is fair to all transitions except the p'-labelled one if the user 'contained in' LSU repeatedly enters his critical section internally.

The infinite trace $p'(p'n)^\omega$ cannot occur in the full ring, since – intuitively speaking – it involves a second access token; the component C in Lemma 6.3 makes sure that there is only one such token.

For the general case, another aspect is even more important: if the congruence were true, it would be not so easy to prove since the nets involved are not safe. We applied our reduction method with partial S-invariants in such a way that only the behaviour of safe nets had to be compared. This is in particular very important if one wants to show congruence with a tool, compare the next subsection.

One could argue that safety can also be obtained by adding a place to each component of the Le-Lann-ring from the beginning, such that Figure 5 shows the modified $LSU[n \to p']$. This way, each component would also accept another access token only if it has given up the previous one before, eliminating the trace we considered above. But this addition would burden the MUTEX-solution unnecessarily; also, reduction would not work directly: the modified components would act as one-place buffers, and a sequence of two such buffers can store two tokens and, thus, does not have the same behaviour as one buffer.

6.2 Dijkstra's Token-Ring

The second MUTEX-solution we want to consider is Dijkstra's token ring [Dij85]; we will describe the first-user view of such a ring analogously to the above. Figure 6 shows the component DU of the first user who owns the access-token on place tok. In Dijkstra's token ring, the user keeps the token when leaving the critical section, so he can use it repeatedly until an order for the token is received (action ro) from the next component; observe the read arc from the now unmarked place no (no order). Then the token is sent (action st) to the next component, say clockwise. Now m is marked, indicating that this component misses the token. If the user requests again, an order is sent (upper so-transition) counter-clockwise to the previous component. Alternatively, the token might have moved on clockwise from the next component such that another order is received from this component (ro) and forwarded counter-clockwise (lower so-transition). If consequently a token is received (rt) from the

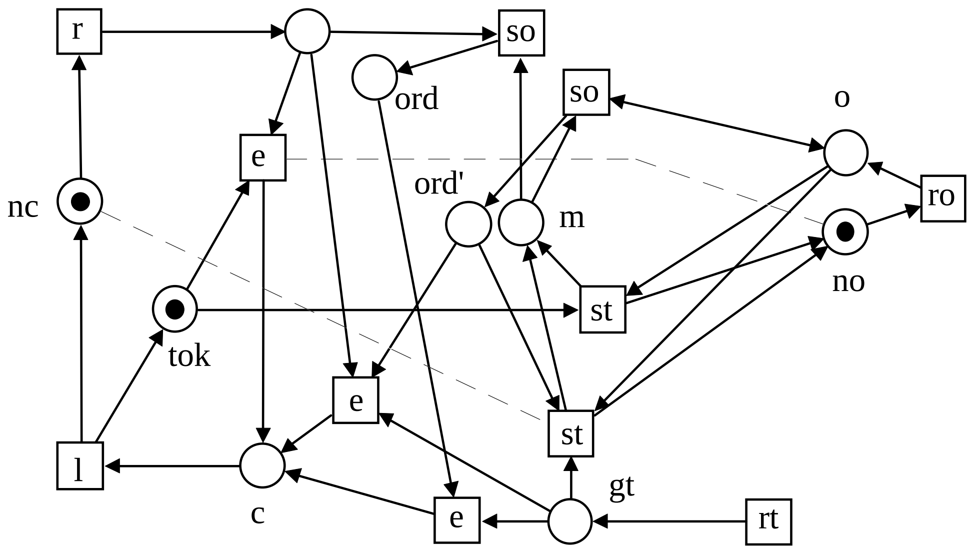

Figure 6

previous component, a request is served in case one is pending or otherwise the token is forwarded (st).

The other components DSU of the first-user view are obtained similarly as above, although this time we keep the actions for the ring communication as they are – in this case so, ro, st and rt. In detail: to get DSU from DU we hide r, e and l and move the token on tok to m; and we duplicate the place nc, but without the read arc and with an internal loop transition instead; compare the black part of Figure 7 below, which has two additional places. Now we define a chain DC_n and a ring DTR_n as in the previous subsection, taking the first user component as chain of length 1 this time.

$$DC_1 = DU$$

$$DC_{n+1} = (DC_n[ro \to o, st \to t] \| DSU[so \to o, rt \to t])/\{o, t\}$$

Such a chain has actions r, e, l, so and rt 'at one end' and ro and st 'at the other end'. We close it to a ring with DSU:

$$DTR_n = (DC_{n-1}[ro \to so, so \to ro, rt \to st, st \to rt] \parallel DSU)/\{so, ro, st, rt\}$$

Again, we will reduce a sequence of components (this time all of type DSU) to a smaller one. An interesting variation is that we have to reduce a sequence of length 3, because reducing 2 components to 1 cannot work: in a sequence of 2 components, the first one can send an order and receive the token, send it to the second component internally and – since it is missing the token now – send another order as the next visible action; thus, it performs $so\ rt\ so$. This is not possible for one component (compare again Figure 7), which after receiving the token must send it on (st) before performing another so.

Since this more complicated application example is difficult to treat by hand, application of a tool is advisable. Since a tool for deciding fair (pre)congruence was not available, a tool for a related precongruence was used; this made some ad-hoc measures necessary as described below.

We make use of two partial S-invariants which we give now. DU is covered by the $(\{rt\},\{st\},1)$-invariant $\{gt,c,tok\}$ making use e.g. of the S-invariant $P = \{c,tok,m,ord,ord'\}$, and it also has the $(\{rt\},\{so\},0)$-invariant $\{gt,c,tok,m\}$. Use of this second partial S-invariant is not really necessary, but it slightly simplifies the behaviour of the nets below, and it makes a little smaller the reachability graphs the tool has to deal with.

Similarly to DU, DSU is covered by an $(\{rt\},\{st\},0)$-invariant, and it also has an $(\{rt\},\{so\},1)$-invariant. With induction, one shows from this that DC_n is covered by an $(\{rt\},\{st\},1)$-invariant and has an $(\{rt\},\{so\},0)$-invariant. A first consequence is then that each DTR_n is safe.

Formally, we will reduce D_3 to D_2 with the following definitions:

$$\begin{aligned} D_2 &= (DSU[ro\to o, st\to t] \parallel DSU[so\to o, rt\to t])/\{o,t\} \\ D_3 &= (D_2[ro\to o, st\to t] \parallel DSU[so\to o, rt\to t])/\{o,t\} \end{aligned}$$

It is not surprising that D_2 (in a composition according to the above partial S-invariants, i.e. with $C = C(\{st\},\{rt\},1)\|C(\{so\},\{rt\},0)$) reacts faster than D_3, and we have verified this with FastAsy according to the notion of faster-than explained e.g. in [BV98]. (For this step, the help of Elmar Bihler is gratefully acknowledged.) As described in [BV98], this implies fair precongruence, where in the present setting we additionally check that the two nets have the same alphabet.

Also, we have essentially verified with FastAsy that D_3 is faster than an elongation of D_2. This elongation concerns the transition that, in the second DSU-component, represents leaving the critical section and the ro-labelled transition; this slows down the reactions of st and so. Together, the verification results imply fair congruence.

More precisely, in order to perform the second verification step with FastAsy, the additional DSU-component of D_3 was replaced by the following net DSU' (– just as the additional LSU-component was transformed in Lemma 6.2). Figure 7 shows in black $DSU \parallel C$; DSU' is DSU with the additional grey transition sc, which is some shortcut, hence the full figure shows $DSU' \parallel C$. The following lemma demonstrates the correctness of this replacement.

Lemma 6.5 *Let $C = C(\{st\},\{rt\},1) \parallel C(\{so\},\{rt\},0)$; then we have $DSU \parallel C =_{\mathcal{FF}} DSU' \parallel C$.*

Proof: For the following arguments, observe that $DSU \parallel C$ is safe due to the coverage by the partial S-invariant given above.

In a firing sequence w of $DSU' \parallel C$ that is fair to all internal transitions, we can replace each occurrence of sc by the three internal transitions corresponding to the r-labelled, the middle e-labelled and the l-labelled transitions of DU. Observe that any transition that is disabled while sc fires is also disabled while these three transitions fire. Hence, if w is fair to some transition t with label in $X \cup \{\lambda\}$ for some X, then the transformed sequence is fair to t as well. Thus, $\mathcal{FF}(DSU' \parallel C) \subseteq \mathcal{FF}(DSU \parallel C)$.

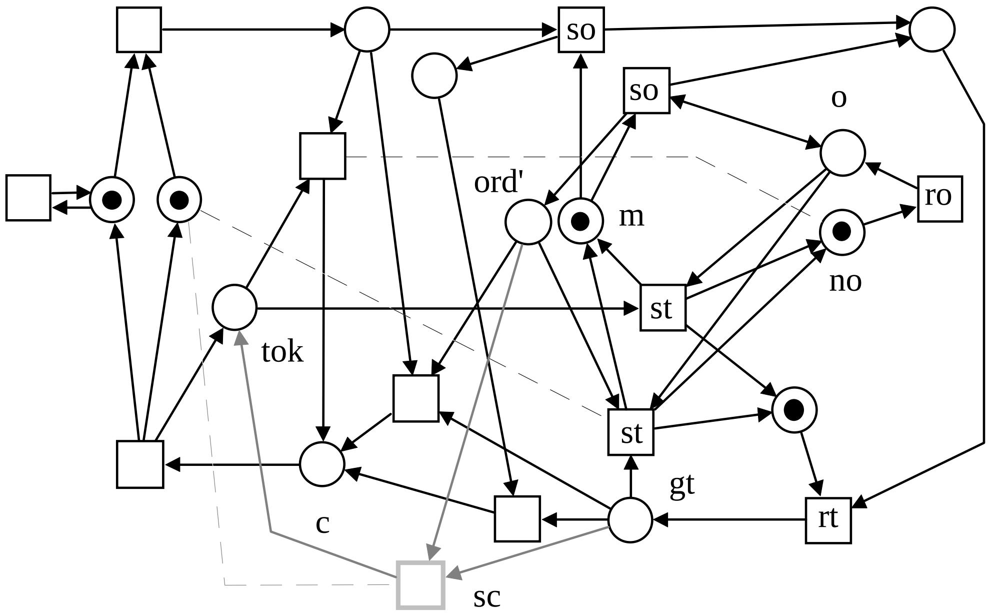

Figure 7

Now consider a firing sequence w of $DSU \,\|\, C$ that is fair to all internal transitions. This w can also fire in $DSU' \,\|\, C$; the only problem could be that it might not be sc-fair, i.e. gt, ord' and nc are permanently marked from some stage onward. Assume this to be the case.

When ord' receives a token for the last time, o is marked. Since a token on ord' implies that m is empty due to the S-invariant P, none of the st-transitions fires, thus o stays marked from this stage onward. Hence, w in $DSU \,\|\, C$ is not fair for the lower st-transition, and st is not refused. (All other actions can be refused, as one sees quite easily.)

If we insert sc (or the three internal transitions as above) into w after the stage mentioned and get w', then tok gets marked in w' while m, ord and ord' stay permanently empty due to P. Thus, w' is fair for all so-, ro-, st-, rt- or λ-labelled transitions, and it gives rise to the same pairs in $\mathcal{FF}(DSU' \,\|\, C)$ as w does in $\mathcal{FF}(DSU \,\|\, C)$. □

As described above, we have obtained the following lemma – using Lemma 6.5 and FastAsy:

Lemma 6.6 *With C as above, $D_3 \,\|\, C =_{\mathcal{FF}} D_2 \,\|\, C$.*

We now proceed as in the proof of Theorem 6.4: We unfold DTR_{n+3} by definition such that we can isolate D_3 in it and then apply the reduction that is possible due to the above partial S-invariants and Lemma 6.6. Thus, we can show

$$\begin{aligned} DTR_{n+3} &= (DC_n[ro \to so, so \to ro, rt \to st, st \to rt] \,\|\, D_3)/\{so, ro, st, rt\} \\ &=_{\mathcal{FF}} (DC_n[ro \to so, so \to ro, rt \to st, st \to rt] \,\|\, D_2)/\{so, ro, st, rt\} \\ &= DTR_{n+2}. \end{aligned}$$

This way, we can reduce rings with at least 4 components to DTR_3; DTR_3 and DTR_2 were shown to satisfy MUTEX-liveness in [BV98]. This finishes our second application:

Theorem 6.7 *For $n > 3$, $DTR_n =_{\mathcal{FF}} DTR_{n-1}$. For $n > 1$, Dijkstra's token ring is a correct MUTEX-solution.*

7 Conclusion and Related Literature

In this paper, we have defined partial S-invariants for a setting where nets are composed by merging transitions – modelling a parallel composition with synchronous communication. We have shown how to derive from partial S-invariants of the components (partial) S-invariants and safety of composed nets. As already mentioned, this is analogous to the partial S-invariants of [Kin95] for a setting where nets are composed by merging places. More importantly, we have shown how partial S-invariants give rise to interface descriptions that can be useful in compositional reduction.

We have applied this idea to show the correctness of two families of MUTEX-solutions based on token rings – exploiting also a vital symmetry result from [BV98]. In this application, we have shown that two (three resp.) components of these rings are equivalent to one (two resp.) component(s) in the proper context, which is derived from partial S-invariants . (The reduction of three to two components was verified using the tool FastAsy, which compares performance of asynchronous systems.) Thus, for each of the two families each member is equivalent to a small member; hence, correctness of this small net shown in [BV98] carries over to the full family. This approach is called behavioural fixed-point approach in [VK98]. The equivalence we have used is the coarsest congruence for the operators of interest respecting fair behaviour in the sense of progress assumption.

Interface descriptions for compositional reduction of some system $N\|N'$ have been studied in [GSL96]: there, some interface description is guessed, used in the reduction of N' to N'' and then verified during the further reduction of $N\|N''$. The last step considers the behaviour of N, and this is not feasible in the fixed-point approach where N is the rest of the ring, i.e. not fixed. In contrast, partial S-invariants give verified interface descriptions; we have derived them with induction from a syntactic inspection of the ring components, i.e. without considering their behaviour by constructing a reachability graph and possibly encountering a state explosion. The latter points out the potential use of partial S-invariants in compositional reduction in general.

The fixed-point approach is very similar to the approach in [WL89]: there, a preorder is used instead of an equivalence; an invariant (or representative) process I has to be found manually, and then it is checked that P is less than I and that, whenever some Q is less than I, $P\|Q$ is less than I. This implies that the composition of any number of components P is less than I; with a suitable preorder and I, this implies that some relevant properties hold for all these compositions. As an example, MUTEX-safety of a version of Le Lann's ring is shown. More generally, [CGJ95] considers networks that might be built from various different types of components according to a network grammar, and a representative for each type is used. As an example, MUTEX-safety of Dijkstra's token ring is shown.

Considering the construction of the complete system from its components, the verification of the above representatives is bottom-up ignoring the context. Determining useful partial S-invariants exactly looks at this context; how this can be done in the approach of [WL89] and [CGJ95] deserves further consideration. As Proposition 5.2 demonstrates, we could also work with the preorder $\leq_{\mathcal{FF}}$ instead of the equivalence $=_{\mathcal{FF}}$.

An important point is that the referenced papers use labelled transition systems while we use Petri nets that in themselves are usually smaller; this is in particular an advantage when determining partial S-invariants. Also, they allow to consider the subtleties of the progress assumption (i.e. the difference between loops and read arcs). Hence, we use a behavioural equivalence that takes the progress assumption into account, in contrast to the above papers that use failure-inclusion [WL89], a refinement thereof [VK98] or a simulation preorder [CGJ95]. Liveness properties often only hold under the progress assumption.

It is planned to verify also the third family of MUTEX-solutions considered in [BV98]. The problem is that in this family data from an infinite set are used (unique identifiers for the components), and it has to be checked whether these systems can be considered as data-independent in some sense. An interesting case study involving data-independence and a reduction as in [WL89] and [CGJ95] can be found in [Kai97].

References

[BBCP00] P. Baldan, N. Busi, A. Corradini, and M. Pinna. Functional concurrent semantics for Petri nets with read and inhibitor arcs. In C. Palamidessi, editor, *CONCUR 2000*, Lect. Notes Comp. Sci. 1877, 442–457. Springer, 2000.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31:560–599, 1984.

[BV98] E. Bihler and W. Vogler. Efficiency of token-passing MUTEX-solutions – some experiments. In J. Desel et al., editors, *Applications and Theory of Petri Nets 1998*, Lect. Notes Comp. Sci. 1420, 185–204. Springer, 1998.

[CGJ95] E.M. Clarke, O. Grumberg, and S. Jha. Verifying parameterized networks using abstraction and regular languages. In I. Lee and S. Smolka, editors, *CONCUR 95*, Lect. Notes Comp. Sci. 962, 395–407. Springer, 1995.

[Dij85] E.W. Dijkstra. Invariance and non-determinacy. In C.A.R. Hoare and J.C. Sheperdson, editors, *Mathematical Logic and Programming Languages*, 157–165. Prentice-Hall, 1985.

[Fra86] N. Francez. *Fairness.* Springer-Verlag, 1986.

[Gol88] R. Gold. Verklemmungsfreiheit bei modularer Konstruktion fairer Petrinetze. Diplomarbeit, Techn. Univ. München, 1988.

[GSL96] S. Graf, B. Steffen, and G. Lüttgen. Compositional minimisation of finite state systems using interface specifications. *Formal Aspects of Computing*, 8:607–616, 1996.

[JK95] R. Janicki and M. Koutny. Semantics of inhibitor nets. *Information and Computation*, 123:1–16, 1995.

[Kai97] R. Kaivola. Using compositional preorders in the verification of sliding window protocol. In *CAV 97*, Lect. Notes Comp. Sci. 1254, 48–59. Springer, 1997.

[Kin95] E. Kindler. *Modularer Entwurf verteilter Systeme mit Petrinetzen.* PhD thesis, Techn. Univ. München, Bertz-Verlag, 1995.

[MR95] U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32:545–596, 1995.

[Pet81] J.L. Peterson. *Petri Net Theory.* Prentice-Hall, 1981.

[Rei84] W. Reisig. Partial order semantics versus interleaving semantics for CSP-like languages and its impact on fairness. In J. Paredaens, editor, *Automata, Languages and Programming*, Lect. Notes Comp. Sci. 172, 403–413. Springer, 1984.

[Rei85] W. Reisig. *Petri Nets.* EATCS Monographs on Theoretical Computer Science 4. Springer, 1985.

[VK98] A. Valmari and Kokkarinen. Unbounded verification results by finite-state compositional technique: 10^{any} states and beyond. In *Int. Conf. Application of Concurrency to System Design, 1998, Fukushima, Japan*, 75–87. IEEE Computer Society, 1998.

[Vog92] W. Vogler. *Modular Construction and Partial Order Semantics of Petri Nets.* Lect. Notes Comp. Sci. 625. Springer, 1992.

[Vog95] W. Vogler. Fairness and partial order semantics. *Information Processing Letters*, 55:33–39, 1995.

[Vog97] W. Vogler. Efficiency of asynchronous systems and read arcs in Petri nets. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *ICALP 97*, Lect. Notes Comp. Sci. 1256, 538–548. Springer, 1997. Full version at http://www.informatik.uni-augsburg.de/∼vogler/ under the title 'Efficiency of Asynchronous Systems, Read Arcs, and the MUTEX-Problem'.

[VSY98] W. Vogler, A. Semenov, and A. Yakovlev. Unfolding and finite prefix for nets with read arcs. In D. Sangiorgi and R. de Simone, editors, *CONCUR 98*, Lect. Notes Comp. Sci. 1466, 501–516. Springer, 1998. Full version as Technical Report Series No. 634, Computing Science, University of Newcastle upon Tyne, February 1998; can be obtained from: ftp://sadko.ncl.ac.uk/pub/incoming/TRs/.

[WL89] P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants. In *Automatic Verification Methods for Finite Systems*, Lect. Notes Comp. Sci. 407, 68–80. Springer, 1989.