Distributed Unfolding of Petri Nets*

Paolo Baldan[1], Stefan Haar[2], and Barbara König[3]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy
[2] INRIA Rennes, *Distribcom* team, France
[3] Institut für Formale Methoden der Informatik, Universität Stuttgart, Germany

Abstract. Some recent Petri net-based approaches to fault diagnosis of distributed systems suggest to factor the problem into local diagnoses based on the unfoldings of local views of the system, which are then correlated with diagnoses from neighbouring supervisors. In this paper we propose a notion of system factorisation expressed in terms of pullback decomposition. To ensure coherence of the local views and completeness of the diagnosis, data exchange among the unfolders needs to be specified with care. We introduce interleaving structures as a format for data exchange between unfolders and we propose a distributed algorithm for computing local views of the unfolding for each system component. The theory of interleaving structures is developed to prove correctness of the distributed unfolding algorithm.

1 Introduction

Partial order semantics are often instrumental in providing a compact representation of the behaviour of concurrent systems: modelling concurrency of events in an explicit way rather than considering all the possible interleavings of such events helps in tackling the so-called state explosion problem. In recent years there has been a growing interest in the use of unfolding-based approaches. Originally introduced in the setting of Petri nets, the unfolding semantics [14] is a branching partial order semantics which represents in a single structure all the possible events in computations, and their causal dependencies and conflicts (branching points). Each branch represents a concurrent execution of the net, in the form of an acyclic conflict-free net. The unfolding provides an "efficient" representation of the state space of the system, not only taking advantage of a partial order representation of concurrency, but also keeping together different computations in its branching structure until a conflict is reached.

When analysing a complex system it happens frequently that we want to consider only a small part of such a system: either because the system is inherently distributed and each observer has access only to a local component of it, or because the system is too big to be analysed or monitored as a whole. In this paper, we take Petri nets as systems models and their unfolding semantics as reference

* Partially supported by EC RTN 2-2001-00346 SegraVis, MIUR project PRIN 2005015824 ART, European NoE ARTIST (IST-2001-34820), French RNRT project SWAN (No. 03 S 481), DFG project SANDS, SFB 627 (NEXUS).

L. Aceto and A. Ingólfsdóttir (Eds.): FOSSACS 2006, LNCS 3921, pp. 126–141, 2006.

semantics, and we view systems as made up of smaller components. Then we show how the projection of the semantics of the whole system over each single local component can be computed locally via a distributed unfolding algorithm, requiring only a minimal interaction among components.

The original motivation of this work is not verification, but distributed diagnosis of asynchronous systems. The general principle of diagnosis for discrete event systems (DES) can be stated as follows: not all transitions of a system are observable; in particular, faults are invisible and have to be deduced from the observations. This deduction of behaviour from the observations is the topic of diagnosis; efforts to force the system into a desired behaviour are studied in the domain of *control*. Although it has an important intersection with diagnosis, control is a clearly distinct problem, not addressed by the present article. Diagnosis can be approached via the construction of finite automata, the *diagnosers*, detailed in [19]; the input of a diagnosis procedure is the language of observed sequences and its output is the language of behaviours that explain the observations. Communication among diagnosers allows for a decentralised diagnosis, in which different diagnoses are proposed by various local diagnosers and then merged to filter out incompatible local views (see, e.g., [6, 16, 5]). Some authors consider timed extensions [17] or use Petri nets as system models [10].

The diagnosis approach in [3, 4, 9], which the present work builds upon, differs from all of the above in the fact that the asynchronous behaviour is captured by partial order semantics, thus abstracting away time aspects and interleavings of concurrent events in order to fight state space explosion. The system behaviour is given in the form of a Petri net model, where only a subset of transitions is observable. Then a sequence (or partially ordered scenario) of observations, called *alarm pattern*, is explainable by several net computations. These explanations are obtained by unfolding the synchronous product of the model net with the alarm pattern, and extracting the maximal configurations compatible with the alarm pattern (see [3]). This approach suffers, for large systems, from the explosion of the size of the global unfolding. Moreover, the practice in diagnosis for large networks justifies the use of several supervisors having only a partial view of the network.

This leads to the idea of *distributed diagnosis via unfoldings*: each supervisor computes a local diagnosis and an exchange of messages with neighbouring supervisors allows to eliminate branches that do not appear as local traces of admissible global configurations. Being able to construct the local views of the global unfolding (of prohibitive size in general) without computing it is the heart of the problem. We remark that we are interested in the projections over the local components of the unfolding of the whole system rather than in the unfoldings of the components themselves. This will become clearer in the technical treatment, but intuitively the reason is that the "autonomous" unfolding of each single component would include "spurious" runs which, although consistent with the structure of the local component itself, have no counterpart in the behaviour of the whole system, due to component interactions. In [4, 9] Petri net components were fused on places, and the fusion of views was done through products

of event structures obtained by using a projection operation with an exchange of messages relating transition actions. Another fusion approach using so-called *augmented processes* is developed in [7, 8].

At a technical level, we will introduce a decomposition/composition mechanism based on pullbacks which allows to view a given Petri net N_3 as built as the join of two components N_1 and N_2 (or more) along a common interface net N_0. The categorical approach allows to exploit a compositionality result which plays a basic role in the design of the distributed unfolding algorithm: the unfolding construction can be expressed as a right adjoint functor between suitable categories of nets and thus it preserves pullbacks. Hence the unfolding of a net N_3, arising as the the pullback of N_1 and N_2 along N_0, can be obtained as the pullback of the unfoldings of the single components.

In order to compute the projections of the full unfolding over the various net components, we propose a distributed algorithm requiring an exchange of information among such components. The components communicate through the interface net, whose unfolding is used to store information about dependencies on events induced by both components. This information is conceptually stored in so-called interleaving structures, whose theory provides a solid theoretical basis for proving the correctness of the distributed unfolding procedure. More specifically, factorisation results from category theory will be used to show that the information stored in the interface suffices in order to obtain the desired result.

The paper is organised as follows. In §2 we lay some general technical ground for the categorical techniques involved. In §3 we focus on Petri net decomposition, while in §4 we introduce Petri net unfoldings. In §5 we develop the theory of interleaving structures, which play a basic role in the distributed unfolding algorithm presented in §6. Finally, in §7 we draw some conclusions.

2 Notation and Categorical Background

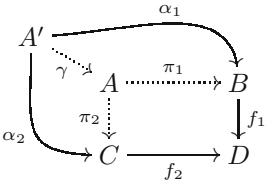
Given a (possibly partial) function $f : A \dashrightarrow B$ and $a \in A$ we will write $f(a) \downarrow$ whenever f is defined on a and $f(a) \uparrow$, otherwise.

Let A be a set. The powerset of A is denoted by $\mathbf{2}^A$. A *multiset* of A is a total function $M\colon A \to \mathbb{N}$. It is called *finite* if the underlying set $\{a \in A \mid M(a) > 0\}$ is finite. A finite multiset is sometimes denoted as a formal sum $M = \bigoplus_{a\in A} M(a) \cdot a$. The set of finite multisets of A is denoted by μA. A subset $X \subseteq A$ will be often treated as the multiset $\bigoplus_{a\in X} 1 \cdot a$.

A *(finitary) multirelation* $f : A \leftrightarrow B$ is a multiset of $A \times B$ such that for all $a \in A$ the set $\{b \in B \mid f(a, b) > 0\}$ is finite. Any multirelation $f : A \leftrightarrow B$ induces a function $\mu f : \mu A \to \mu B$ defined by $\mu f(\bigoplus_{a\in A} n_a \cdot a) = \bigoplus_{b\in B}(\sum_{a\in A} n_a\, f(a, b)) \cdot b$. We say that a multirelation $f : A \leftrightarrow B$ is *total* if for any $a \in A$ there exists $b \in B$ such that $f(a, b) > 0$, *injective* if for any $b \in B$ we have $\sum_{a\in A} f(a, b) \leq 1$, *surjective* if for any $b \in B$ we have $\sum_{a\in A} f(a, b) \geq 1$.

We will refer to some categorical concepts (see also [1]), and in particular we will make extensive use of pullbacks and factorisation structures.

Definition 1 (pullback). *Let* $\mathbf{C}$ *be a category and* $f_1 : B \to D$, $f_2 : C \to D$ *be arrows in* $\mathbf{C}$. *The* pullback *of* f_1 *and* f_2 *is an object* A *(*pullback object*) and a pair of arrows* $\pi_1 : A \to B$, $\pi_2 : A \to C$ *such that (i)* $f_1 \circ \pi_1 = f_2 \circ \pi_2$ *and (ii) for any object* A' *with arrows* $\alpha_1 : A' \to B$, $\alpha_2 : A' \to C$ *such that* $f_1 \circ \alpha_1 = f_2 \circ \alpha_2$ *there exists a unique arrow* $\gamma : A' \to A$ *such that* $\pi_i \circ \gamma = \alpha_i$ *(*$i \in \{1, 2\}$*).*



For instance, for a fixed a set Λ of labels, consider the category $\mathbf{LSet}^*$ of Λ-labelled sets and partial functions. Objects are pairs (A, λ), where A is a set and $\lambda\colon A \to \Lambda$ is a total labelling function, while arrows are label-preserving partial functions. Given two arrows $f_1\colon (B, \lambda_B) \to (D, \lambda_D)$, $f_2\colon (C, \lambda_C) \to (D, \lambda_D)$ the pullback object is (A, λ_A) with

$$\begin{aligned} A = {} & \{(b, c) \mid b \in B, c \in C, f_1(b) = f_2(c)\} \\ & \cup \{(b, *) \mid b \in B, f_1(b) \uparrow\} \cup \{(*, c) \mid c \in C, f_2(c) \uparrow\} \\ & \cup \{(b, c) \mid b \in B, c \in C, f_1(b), f_2(c) \uparrow \text{ and } \lambda_B(b) = \lambda_C(c)\} \end{aligned}$$

and λ_A, π_1 and π_2 defined in the obvious way.

Definition 2 (factorisation structures). *Let* $\mathbf{C}$ *be a category and let* E, M *be classes of morphisms in* $\mathbf{C}$*, closed under composition with isomorphisms. The pair* (E, M) *is called a* factorisation structure *for morphisms in* $\mathbf{C}$ *and* $\mathbf{C}$ *is called* (E, M)-structured *whenever*

- $\mathbf{C}$ *has* (E, M)-factorisations *of morphisms, i.e., each morphism* f *of* $\mathbf{C}$ *has a factorisation* $f = m \circ e$ *with* $e \in E$ *and* $m \in M$.
- $\mathbf{C}$ *has the unique* (E, M)-diagonalisation property*, i.e., for each commutative square as shown on the left-hand side below with* $e \in E$ *and* $m \in M$ *there exists a unique* diagonal*, i.e., a morphism* d *such that the diagram on the right-hand side commutes (i.e., such that* $d \circ e = f$ *and* $m \circ d = g$*).*

The classical example of (E, M)-factorisation in **Set** is the factorisation of a function f into a surjective and an injective part. In the following, morphisms from E are drawn using double-headed arrows of the form $A \twoheadrightarrow B$, whereas morphisms from M are drawn using arrows of the form $A \rightarrowtail B$.

In any (E, M)-structured category (E, M)-factorisations of morphisms are unique up to isomorphism, the sets E and M are both closed under composition and all arrows in M are stable under pullback.

3 Composing Petri Nets

In this section we introduce the basics of Petri nets and the corresponding category. Then we present a technique for decomposing Petri nets into smaller components (or equivalently to compose Petri nets) along a given interface, showing how the operation can be interpreted, in categorical terms, as a pullback.

We will consider labelled Petri nets, with morphisms as introduced in [20]. In the rest of the paper Λ denotes a fixed label set for all considered Petri nets.

Definition 3 (Petri net). *A* Petri net *is a tuple $N = (S, T, \lambda, {}^\bullet(), ()^\bullet, m)$ where S is the set of places, T is the set of transitions, $\lambda: T \to \Lambda$ is a labelling function, ${}^\bullet(), ()^\bullet: T \to \mathbf{2}^S$ associate to each transition $t \in T$ its pre-set and post-set, respectively, and $m \in \mathbf{2}^S$ is the initial marking.*

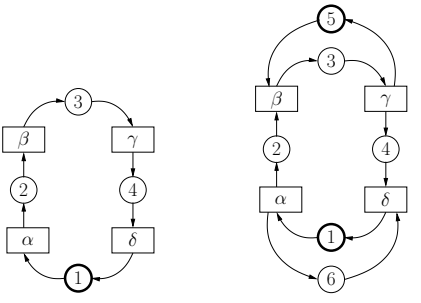
A (Petri net) morphism $\tau = (\eta, \beta): N \to N'$ is a pair consisting of a partial function $\eta: T \dashrightarrow T'$ and a finitary multirelation $\beta: S \leftrightarrow S'$ such that (i) $\mu\beta(m) = m'$, and (ii) for any $t \in T$, $\mu\beta({}^\bullet t) = {}^\bullet\eta(t)$ and $\mu\beta(t^\bullet) = \eta(t)^\bullet$, where conventionally ${}^\bullet\eta(t) = \eta(t)^\bullet = \emptyset$ when $\eta(t) \uparrow$. The category of Petri nets and their morphisms is denoted by **PN**.

In the sequel we will assume that in any considered Petri net, all transitions have a non-empty pre-set, a typical property required in unfolding-based approaches. Moreover we will denote the components of a Petri net N as S, T, λ, ${}^\bullet()$, $()^\bullet$ and m. Superscripts will carry over to the component names.

Example: Examples of Petri nets can be found in Fig. 1. Initially marked places are drawn with thick lines. Both nets consist of a loop involving four transitions, labelled over the set $\Lambda = \{\alpha, \beta, \gamma, \delta\}$.

For defining formally the local projections of the full unfolding we need some special classes of morphisms.

Definition 4 (projection and embedding). *A Petri net morphism $\tau = (\eta, \beta) : N \to N'$ is called a* projection *whenever η and β are surjective. It is called an* embedding *if both η and β are total and injective.*



(a) N'_3 (b) N_3

Fig. 1. Two examples of Petri nets

It can be shown that **PN** is (projection,embedding)-structured (*pe*-structured, for short). Given a **PN** morphism $\tau = (\eta, \beta) : N \to N'$, let $\tau(N)$ denote the subnet of N' including only transitions in $\eta(T)$. Then the projection $\tau : N \to \tau(N)$ and the inclusion of $\tau(N)$ into N' provide a *pe*-factorisation of τ.

In the following we define how to restrict a Petri net to a subset S_0 of its places. Specifically, a transition t will appear in the new net only if it is connected to at least one place in S_0.

Definition 5 (restricting a net). *Let N be a net and let $S_0 \subseteq S$ be a subset of places. Then the* restriction *of N to S_0, denoted $[N]_{S_0} = (S_0, T_0, \lambda_0, {}^\bullet(), ()^\bullet, m_0)$, is defined as follows: $T_0 = \{(t,0) \mid t \in T \wedge ({}^\bullet t \cap S_0 \neq \emptyset \ \vee \ t^\bullet \cap S_0 \neq \emptyset)\}$, $\lambda_0((t,0)) = \lambda(t)$, ${}^\bullet(t,0) = {}^\bullet t \cap S_0$, $(t,0)^\bullet = t^\bullet \cap S_0$ and $m_0 = m \cap S_0$.*

This induces a morphism $\tau_{N,S} = (\eta, \beta) \colon N \to [N]_S$ with $\eta(t) = (t,0)$, whenever $(t,0) \in T_0$, and $\eta(t) \uparrow$ otherwise. Furthermore $\beta(s,s') = 1$, whenever $s = s' \in S_0$, and $\beta(s,s') = 0$ otherwise.

The next proposition provides a recipe for decomposing Petri nets along some chosen places, which play the role of interface between the subcomponents.

Proposition 6 (decomposition of Petri nets). *Let N_3 be a Petri net and let $S_3 = S_1 \cup S_2$. Let $S_0 = S_1 \cap S_2$ and construct nets N_0, N_1, N_2 as follows: $N_1 = [N_3]_{S_1}$, $N_2 = [N_3]_{S_2}$, $N_0 = [N_1]_{S_0} = [N_2]_{S_0}$. Say that transitions in $T_i - T_0$ are* local *to N_i for $i \in \{1,2\}$ and assume that transitions local to different nets have distinct labels. Then the nets N_i with $i \in \{0,1,2,3\}$ together with the morphisms τ_{N_3,S_1}, τ_{N_3,S_2}, τ_{N_1,S_0}, τ_{N_2,S_0} form a pullback diagram.*

Note that, in order to exploit the results about unfolding, also the component nets must not contain transitions with empty pre-sets. Henceforth, all decompositions are supposed to have this property. For safe nets this can be achieved by introducing extra complement places. Also, decomposition will have to be done in such a way that local transitions in different components have different labels.

Example: Consider the Petri net N_3' in Fig. 1(a). We intend to split the loop along the places 1 and 3, i.e., we plan to decompose as described in Proposition 6 with $S_1 = \{1,2,3\}$ and $S_2 = \{1,3,4\}$. However, this would result in subcomponents N_0, N_1 and N_2 including transitions with empty pre-set. In order to avoid this problem, we can complement the interface places 1, 3 by adding two more places 5, 6, thus obtaining the net N_3 in Fig. 1(b). Call place $\bar{p}$ the *complement* of place p if $p^\bullet = {}^\bullet\bar{p}$, $\bar{p}^\bullet = {}^\bullet p$, and $p \in m \Leftrightarrow \bar{p} \notin m$. Then 5, 6 are complements for 1, 3. The new net is equivalent to N_3' (in a sense which can be formalised [15]) and can be safely decomposed using $S_1 = \{1,2,3,5,6\}$ and $S_2 = \{1,3,4,5,6\}$.

We split N_3 into two subnets N_1, N_2 with interface N_0 (according to Proposition 6), thus obtaining the pullback in category **PN** shown in Fig. 2.

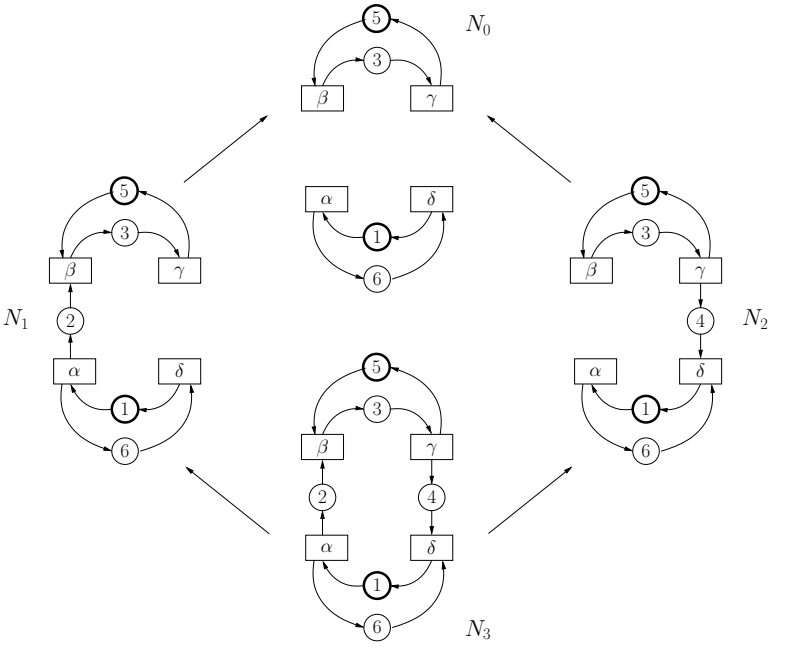


Fig. 2. Decomposing a loop as a pullback of nets

4 Unfolding Petri Nets

Recall that given a Petri net N the dependencies between transitions are captured by two basic relations, causality and conflict. *Causality* is the least transitive relation $<_N$ over $S \cup T$ such that if $s \in {}^\bullet t$ then $s <_N t$ and if $s \in t^\bullet$ then $t <_N s$. We denote by $\leq_N$ the reflexive closure of $<_N$ and for any $x \in S \cup T$, $\lfloor x \rfloor = \{y \in S \cup T : y \leq_N x\}$. *Conflict* is the least symmetric relation $\#_N$ over $S \cup T$ such that (i) if ${}^\bullet t \cap {}^\bullet t' \neq \emptyset$ and $t \neq t'$ then $t \#_N t'$ and (ii) if $t \#_N t'$ and $t <_N t''$ then $t'' \#_N t'$.

Occurrence nets are basically acyclic nets where each place is generated by at most one transition. They are used to unfold Petri nets as described below.

Definition 7 (occurrence net). *An* occurrence net *is a net N satisfying:*

1. *if $t^\bullet \cap t'^\bullet \neq \emptyset$ then $t = t'$;*
2. *$\leq_N$ is a partial order and $\lfloor t \rfloor$ is finite for any $t \in T$;*
3. *the* initial marking *m is the set of $\leq_N$-minimal places;*
4. *$\#_N$ is irreflexive.*

With **ON** *we denote the full subcategory of* **PN** *having occurrence nets as objects.*

A *configuration* of an occurrence net N, formalising the intuitive idea of "concurrent run", is a subset $C \subseteq T$ such that C is left-closed w.r.t. $\leq_N$ and free of

conflicts. A set of places $X \subseteq S$ is called *concurrent*, written $conc(X)$, if $\lfloor X \rfloor$ is a configuration and $\neg(s <_N s')$ for all $s, s' \in X$.

We are now ready to define the unfolding of a Petri net. The unfolding construction unwinds a given net N into an occurrence net, starting from the initial marking, firing transitions in all possible ways and recording the corresponding occurrences. For the sake of presentation we give an equational definition.

Definition 8 (unfolding). *Let N be a Petri net. Its unfolding $\mathcal{U}(N)$ and the* folding morphism $\tau_N = (\eta, \beta) : \mathcal{U}(N) \to N$ *are the occurrence net and net morphism determined by the following recursive equations, where the components of the unfolding are primed:*

$$
\begin{aligned}
&m' = \{\langle \emptyset, s\rangle \mid s \in m\}\\
&S' = m' \cup \{\langle \{t'\}, s\rangle \mid t' = \langle X, t\rangle \in T' \ \wedge \ s \in t^\bullet\}\\
&T' = \{\langle X, t\rangle \mid X \subseteq S' \ \wedge \ conc(X) \ \wedge \ t \in T \ \wedge \ \mu\beta(X) = {}^\bullet t\}\\
&\textit{For } t' = \langle X, t\rangle \in T' : \qquad {}^\bullet t' = X \quad \textit{and} \quad t'^\bullet = \{\langle \{t'\}, s\rangle \mid s \in t^\bullet\}\\
&\eta(t') = t \qquad \text{iff} \ \ t' = \langle X, t\rangle\\
&\beta(s', s) = 1 \ \ \text{iff} \ \ s' = \langle x, s\rangle \quad (x \in \mathbf{2}^{T'})
\end{aligned}
$$

Observe that items in the unfolding are enriched with their causal histories. Place $s' = \langle x, s\rangle$ records its generator x (x is empty when the place is in the initial state, otherwise x is a singleton) and the place s in the original net; transition $t' = \langle X, t\rangle$ represents a firing of t that consumes the resources in X.

Proposition 9 (right adjoint [20, 13]). *The construction $\mathcal{U}$ extends to a functor $\mathcal{U} : \mathbf{PN} \to \mathbf{ON}$, right adjoint to the inclusion of* **ON** *into* **PN**.

Right adjoints preserve limits (see [12, 1]) and hence also pullbacks, which are special limits. As a consequence the unfolding of a pullback in **PN** is the pullback (in **ON**) of the unfoldings of the component nets. More precisely, given a pullback $\tau_i' : N_3 \to N_i$, $\tau_i : N_i \to N_0$ ($i \in \{1, 2\}$) in **PN**, we have that $\mathcal{U}(\tau_i') : \mathcal{U}(N_3) \to \mathcal{U}(N_i)$, $\mathcal{U}(\tau_i) : \mathcal{U}(N_i) \to \mathcal{U}(N_0)$ ($i \in \{1, 2\}$) is a pullback in **ON**. This result will play a central role in the rest of this paper.

Example: Unfolding the nets N_0, N_1, N_2 and N_3 of Fig. 2 we obtain the occurrence nets O_0, O_1, O_2 and O_3 in Fig. 3 (ignore the shaded places and transitions for the moment). Transitions in the occurrence nets are named by using their label, with an additional index. The morphisms to the original nets are the obvious ones suggested by the labelling. By the considerations above, the occurrence net O_3 arising as unfolding of N_3 is the pullback of O_1 and O_2 along O_0.

The aim of this paper is to compute—in a distributed way—the projection of $\mathcal{U}(N_3)$ to $\mathcal{U}(N_i)$, i.e., the local view of component N_i, when taking into account the behaviour of the other component. The intuitive idea of local view is formalised by using factorisations.

It can be shown that, taking projections and embeddings as in Definition 4, the category **ON** is *pe*-structured. The only delicate point is to show that given an

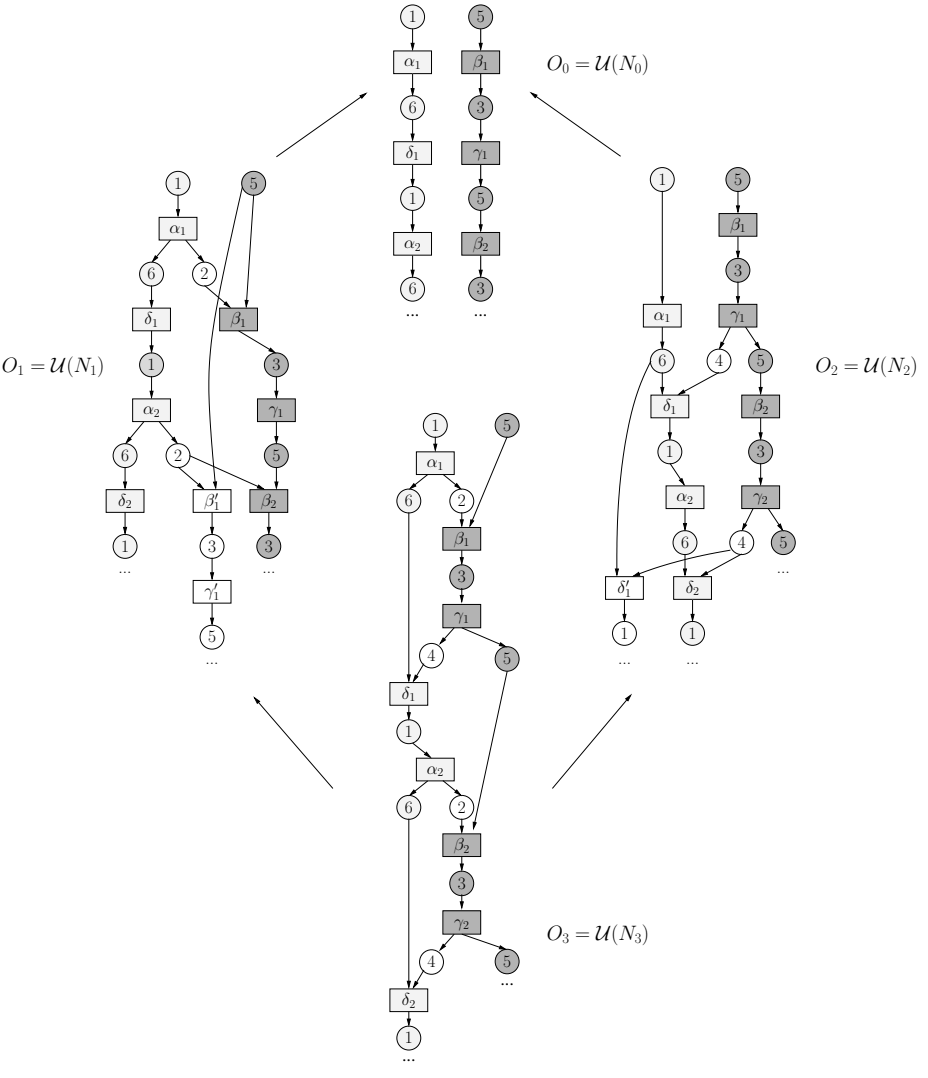


Fig. 3. Composition of unfoldings as pullback of occurrence nets

occurrence net morphism $\tau : O_1 \to O_2$, the net $\tau(O_1)$ as defined in Section 3 is a well-defined occurrence net, but this follows from the fact that occurrence net morphisms reflect causal chains (see [20], Lemma 3.3.6).

Definition 10 (projection of occurrence nets). *Let $\tau = (\eta, \beta) \colon O_1 \to O_2$ be an* **ON** *morphism, and let $\tau^p \colon O_1 \to O_2^1$, $\tau^e \colon O_2^1 \to O_2$ be the* pe*-factorisation of τ. Then the occurrence net O_2^1 is called the* projection *of O_1 onto O_2.*

Example: Consider the unfoldings of our running example in Fig. 3. The shaded places and transitions in O_0, O_1 and O_2 identify the projections O_0^3, O_1^3, O_2^3.

Transitions in O_1 and O_2 which disappear in the projection intuitively represent events that are infeasible if the net components interact. For instance, consider transition β_1' in O_1. From the point of view of N_1, transition δ_1 is a cause for β_1'. However, through the interface, transition β_1' in N_1 corresponds to β_1 in N_2, and in this latter net β_1 is a cause for δ_1. Hence β_1' turns out to be not firable.

5 Interleaving Structures and Their Properties

In order to be able to compute the local projection of the unfolding, intuitively, each net component needs to know the behavioural constraints on the events of the interface net imposed by the other component. Unfortunately, the idea of simply representing dependencies between events, i.e., causality and conflict, with prime event structures, and projecting to the interface the additional dependencies derived in each net component does not work. Consider, for instance, the occurrence nets in Fig. 4, where morphisms φ_i map any transition in N_i to the only transition in the interface net with the same label. Since the two γ-labelled transitions in N_1 are fused in N_0, the projection of causalities in N_1 to N_0 would result in an or-causality between $\{t_0, t_1\}$ and t_2, a phenomenon that is not expressible in a prime event structure. Still, from N_2 we obtain the information that t_2 must be fired before t_1. By combining this knowledge we discover that $t_0 < t_1 < t_2$ is the only possible order in which the transitions of N_0 can be executed. It can be shown that similar problems arise when considering more general partial order models including Winskel's general event structures [20].

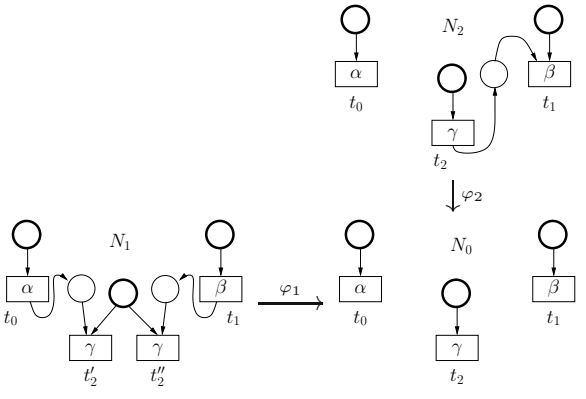


Fig. 4. Projecting dependency relations over the interface

The search for structures suitable to express possible orderings of events and forming a category with nice factorisation properties leads us to so-called interleaving structures. As their name suggests, these structures do not rely on partial orders, but, as discussed in [2], for practical purposes an efficient partial order representation based on occurrence nets can be devised.

Interleavings. For a set A, denote by A^* the set of finite sequences of elements of A and by $A^{\odot}$ the subset of sequences in A^* in which each element occurs at most once. A (partial) function $f : A \dashrightarrow B$ induces a function $f : A^* \to B^*$ (still denoted by f), where for $r \in A^*$ its image $f(r)$ is defined pointwise, removing from the sequence the elements on which f is undefined.

Definition 11 (interleaving structures). *An* interleaving structure *is a tuple $\mathcal{I} = (E, R, \lambda)$ where E is a set of* events*, $\lambda\colon E \to \Lambda$ is a labelling of events and $R \subseteq E^{\odot}$ is the set of* runs*, satisfying: (i) R is prefix-closed, (ii) R contains the empty run ε, and (iii) every event $e \in E$ occurs in at least one run.*

The components of an interleaving structure $\mathcal{I}$ will be denoted by $E_{\mathcal{I}}$, $R_{\mathcal{I}}$, $\lambda_{\mathcal{I}}$, whereas the components of $\mathcal{I}_i$ will also be denoted by E_i, R_i, λ_i.

Definition 12 (interleaving morphisms). *Let $\mathcal{I}_i = (E_i, R_i, \lambda_i)$ with $i \in \{1, 2\}$ be interleaving structures. An* interleaving morphism *from $\mathcal{I}_1$ to $\mathcal{I}_2$ is a partial function $\theta\colon E_1 \dashrightarrow E_2$ on events such that (i) $\lambda_2(\theta(e)) = \lambda_1(e)$ whenever $\theta(e) \downarrow$, and (ii) for every $r \in R_1$ it holds that $\theta(r) \in R_2$. We denote the category of interleaving structures and interleaving morphisms by* **Ilv**.

By Condition (2) above, θ must be injective on the events occurring in any single run. Pullbacks can be constructed in a quite straightforward way in **Ilv**.

Proposition 13 (pullbacks in Ilv). *Let $\theta_i\colon \mathcal{I}_i \to \mathcal{I}_0$, $i \in \{1, 2\}$ be two interleaving morphisms. Their pullback in* **Ilv***, denoted by $\pi_i\colon \mathcal{I}_3 \to \mathcal{I}_i$, $i \in \{1, 2\}$ can be constructed as follows:*

- *Define E'_3 as the pullback in the category of labelled sets and partial functions, and let $\pi'_i\colon E_3 \to E_i$ be the standard partial projections.*
- *Define $R_3 = \{r \in (E'_3)^{\odot} \mid \pi_1(r) \in R_1 \wedge \pi_2(r) \in R_2\}$.*
- *Let $E_3 \subseteq E'_3$ be the subset of events in E'_3 that occur in at least one run in R_3. Furthermore let $\pi_i = \pi'_i|_{E_3}\colon E_3 \to E_i$ be the projections restricted to E_3.*
- *Finally set $\lambda_3((e_1, e_2)) = \lambda_1(e_1) = \lambda_2(e_2)$, $\lambda_3((e_1, *)) = \lambda_1(e_1)$ and $\lambda_3((*, e_2)) = \lambda_2(e_2)$ for all events in E_3.*

Then $\mathcal{I}_3 = (E_3, R_3, \lambda_3)$ is the pullback object.

Factorisation Structures. We next show how to obtain a factorisation structure for **Ilv**. This is needed in order to project information about possible interleavings of events from each component down to the interface, where it can be read by the other component.

Definition 14 (projection, embedding). *An interleaving morphism $\theta\colon \mathcal{I}_1 \to \mathcal{I}_2$ is called* projection *if the induced function on runs $\theta\colon R_1 \to R_2$ is surjective. Morphism θ is called* embedding *if the mapping on events is a total injection.*

Observe that by definition any projection θ is surjective on the set of events.

Given any morphism $\theta\colon \mathcal{I}_1 \to \mathcal{I}_2$ in **Ilv**, a projection-embedding factorisation $\mathcal{I}_1 \overset{\theta^p}{\to} \mathcal{I}_2^1 \overset{\theta^e}{\to} \mathcal{I}_2$ can be obtained by taking as the runs of $\mathcal{I}_2^1$ all runs in $\mathcal{I}_2$ having a preimage under θ, and defining the set of events of $\mathcal{I}_2^1$ and θ^p, θ^e appropriately. The interleaving structure $\mathcal{I}_2^1$ is also called *projection* of $\mathcal{I}_1$ to $\mathcal{I}_2$ via θ.

Proposition 15 (Ilv (E, M)-structured). *The category* **Ilv** *is (E, M)-structured where E is the set of projections and M is the set of embeddings.*

It can be shown that, not only the embeddings, but also the projections are stable under pullbacks in **Ilv**. Note that an analogous proposition does not hold in **ON**. This is one of the reasons for resorting to interleaving structures.

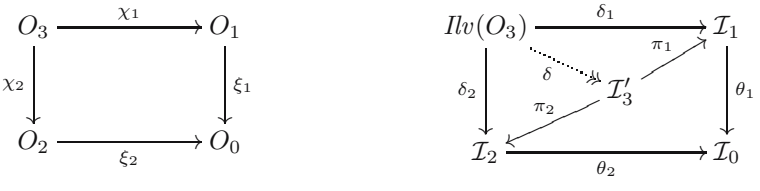
Projections of Interleaving Structures and Occurrence Nets. Every occurrence net O can be associated with an interleaving structure $Ilv(O)$ whose set of events coincides with the set of transitions of the net.

Definition 16. *Let $O = (S, T, \lambda, {}^\bullet(), ()^\bullet, m)$ be an occurrence net. Its interleaving structure is $Ilv(O) = (T, R, \lambda)$, where R consists of all runs $r \in T^\odot$ such that for every prefix r' of r the events occurring in r' form a configuration of O.*

In the following an element $r \in R_{Ilv(O)}$ will be called a run of O.

The mapping *Ilv* can be extended to a functor $Ilv : \mathbf{ON} \to \mathbf{Ilv}$. It can be seen that *Ilv* does not preserve pullbacks, but still a useful relation can be established between pullbacks in **ON** and in **Ilv**.

Lemma 17. *Consider a pullback diagram in* **ON** *as shown in the left-hand side below and take its image through the Ilv functor, thus obtaining the outer square in the right-hand diagram below. Furthermore let $\mathcal{I}'_3$ be the pullback in* **Ilv** *of θ_1 and θ_2. Then the mediating morphism $\delta\colon Ilv(O_3) \to \mathcal{I}'_3$ is a projection.*

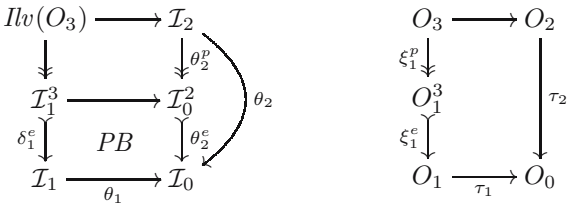


Summing up, we obtain a procedure for determining the projection of a pullback object in **ON** without actually constructing the pullback.

Proposition 18. *Let $\tau_i\colon O_i \to O_0$, $i \in \{1, 2\}$ be two occurrence net morphisms and let $\xi_i\colon O_3 \to O_i$, $i \in \{1, 2\}$ be their pullback. Then the projection O_1^3 and the morphism $O_1^3 \to O_1$ can be determined (without computing O_3) as follows:*

- *Determine the interleaving structures $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2$ corresponding to O_0, O_1, O_2, i.e., $\mathcal{I}_i = Ilv(O_i)$, including their morphisms $\theta_i = Ilv(\tau_i)\colon \mathcal{I}_0 \to \mathcal{I}_i$, $i \in \{1, 2\}$.*
- *Compute the projection-embedding factorisation $\mathcal{I}_2 \overset{\theta_2^p}{\to} \mathcal{I}_0^2 \overset{\theta_2^e}{\to} \mathcal{I}_0$ of θ_2.*
- *Take the pullback of θ_1 and θ_2^e and obtain the morphism $\delta_1^e\colon \mathcal{I}_1^3 \to \mathcal{I}_1$.*
- *Now take the subnet of O_1 that contains the transitions in the image of δ_1^e.*

This gives the projection O_1^3 of O_3 to O_1 with morphism $\xi_1^e : O_1^3 \to O_1$.



6 An Algorithm for Distributed Unfolding

We can now present a distributed unfolding algorithm based on interleaving structures. The algorithm takes as input a pair of net morphisms $\tau_i \colon N_i \to N_0$, $i \in \{1,2\}$ obtained by decomposing a Petri net N_3 as in Proposition 6. Then it builds, in a stepwise fashion, the remaining morphisms of the commuting diagram in Fig. 5, where O_i^j is the projection of $\mathcal{U}(N_j)$ over $\mathcal{U}(N_i)$. When $\xi_i = (\eta_i, \beta_i)$, we will sometimes write $\xi_i(t)$ instead of $\eta_i(t)$.

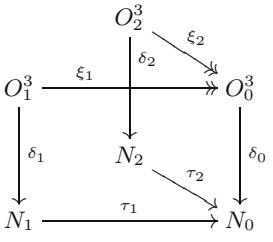


Fig. 5. Nets and morphisms involved in Algorithm 19

Algorithm 19 (distributed unfolding). Denote intermediate states of the occurrence nets and morphisms by $\bar{O}_0$, $\bar{O}_1$, $\bar{O}_2$, $\bar{\xi}_i$, $\bar{\delta}_j$. Start with occurrence nets corresponding to the initial places of N_0, N_1, N_2 and the appropriate corresponding morphisms. At any step transform the morphisms $\bar{\xi}_i$, $\bar{\delta}_i$ as follows: let $j \in \{1,2\}$

(1) Look for a concurrent subset of places X in $\bar{O}_j$ such that $\bar{\delta}_j(X)$ is the pre-set of a transition t in N_j and furthermore[1]
    (*) there exists a run r of $\bar{O}_j$ that contains all causes of X and no consequences of X with $\bar{\xi}_j(r) \in \bar{\xi}_{3-j}(R_{Ilv(\bar{O}_{3-j})})$.
(2) Add $t' = \langle X, t\rangle$ with postset $\{\langle\{t'\}, s\rangle \mid s \in t^\bullet\}$ to $\bar{O}_j$;
    Update $\bar{\delta}_j$ by adding $t' \mapsto t$ and $\langle\{t'\}, s\rangle \mapsto s$.
(3) If $\tau_j(t) = t_0$ is defined,
    add a new transition $t_0' = \langle \bar{\xi}_j(X), t_0\rangle$ with post-set $\{\langle\{t_0'\}, s_0\rangle \mid s_0 \in t_0{}^\bullet\}$ to $\bar{O}_0$, unless it is already present;
    Update $\bar{\delta}_0$ by adding $t_0' \mapsto t_0$ and $\langle\{t_0'\}, s_0\rangle \mapsto s_0$;
    Update $\bar{\xi}_j$ by adding $t' \mapsto t_0'$ and $\langle\{t'\}, s\rangle \mapsto \langle\{t_0'\}, \tau_j(s)\rangle$.

[1] Condition (*) basically states that transition t can be fired after a run r of O_j and this run r is consistent with the behaviour of the other component. That is, there is a way to synchronise r and some run of O_{3-j}.

Assuming that there are two unfolders and a third process which manages the interface information (i.e., which records the projections of the runs of both components) then the checking of Condition (*) and step (3) are performed by unfolder j together with the interface manager, whereas the remaining steps can be performed by unfolder j on its own. Hence communication between unfolders 1 and 2 is restricted to communication via the interface manager.

A transition t in the occurrence net $\bar{O}_0$ is called *valid* if it appears in one of the runs of $R = \bar{\xi}_1(R_{Ilv(\bar{O}_1)}) \cap \bar{\xi}_2(R_{Ilv(\bar{O}_2)})$. A transition t' of $\bar{O}_j$ for $j \in \{1, 2\}$ is *valid* if $\bar{\xi}_j(t') \uparrow$ or there is a run rt' in $\bar{O}_j$ such that $\bar{\xi}_j(rt') \in R$. Note that the algorithm will never generate a transition having a non-valid cause. Furthermore transitions of $\bar{O}_0$ might at some point not be valid but become valid at a later stage when corresponding pre-images have been generated by both unfolders.

Example: The above algorithm, applied to our running example, produces the shaded subparts of the nets in Fig. 3. For instance transition β_1' will never be added to $\bar{O}_1$. This transition may follow the run $\alpha_1\delta_1\alpha_2$, but there is no run r in O_2 for which we have $\xi_2(r) = \alpha_1\delta_1\alpha_2 = \xi_1(\alpha_1\delta_1\alpha_2)$.

In order to ensure that every enabled transition will eventually be chosen, the algorithm unfolds breadth-first: the sets X computed in step (1) of one round have to be worked out completely before those from the next round.

Proposition 20 (correctness of distributed unfolding). *Let $\bar{O}_0$, $\bar{O}_1$ and $\bar{O}_2$ denote the (infinite) unions of the sequences of nets produced by the algorithm above. By restricting $\bar{O}_0$, $\bar{O}_1$, $\bar{O}_2$ to the valid transitions (and their pre- and post-sets plus the initial places), one obtains exactly the occurrence nets O_0^3, O_1^3, O_2^3, where O_i^j is the projection of $\mathcal{U}(N_j)$ over $\mathcal{U}(N_i)$.*

7 Conclusion

We have presented a distributed algorithm for Petri net unfoldings based on pullback decompositions, whose use allows to factor the global unfolding into local views. In fact, computation of the—potentially large—global unfolding of a distributed system is avoided; local supervisors develop their local views, guided by message exchange with their peers through an interface net. As a data structure for this communication, event structures would appear as a natural choice, but for all considered branches of event structures (e.g., prime, bundle, stable, general event structures) important properties concerning factorisations and projections were lacking. This difficulty has been overcome by introducing the category of interleaving structures, which has been shown to enjoy the needed properties. The investigation of partially ordered models and related categories for the correlation of local views is a theme for future investigation. Some results concerning partial order representations for interleaving structures can be found in [2].

We gave a distributed unfolding algorithm in the case of two peers interacting through an interface. This calls for a generalisation to an arbitrary number of peers and unfolders. If all components share the same interface, this generalisation is straightforward: we only have to replace pullbacks by so-called

wide pullbacks of diagrams with several arrows having a common target object. The case where, for instance, the system consists of three components, and the interface between components 1 and 2 is different from the interface between components 1 and 3 is not straightforward and represents a matter of future investigation.

The task we addressed is closely related to that of [4, 9], so the differences deserve to be pointed out. A first one resides in the notion of system factorisation: [4, 9] use a composition operation between Petri nets based on place fusion, so transition occurrences have to be communicated between components and a sophisticated label coding is used to determine the local effect of a transition. Our approach essentially relies on a composition operation along an explicit interface, formalised as a pullback in a suitable category of nets; in the pullback decomposition, transitions acting on shared places are necessarily shared themselves. This contributed to making the algorithm simpler and easier to understand. Moreover, moving from the (computationally hard) products of event structures used in [4, 9] to the pullback of interleaving structures (possibly computed through their partial order representation) can lead to a gain in efficiency for the algorithm. More generally, the fact that our approach is developed in a categorical setting suggests a way for adapting it to different computational models, e.g., variations of Petri nets or more expressive models, like graph transformation systems [18]. This will only require to verify that the needed properties are satisfied by the category of models at hand.

Finally, distributed unfolding is orthogonal to the parallelisation of Petri net unfoldings in [11]: that work parallelises the computation of the global unfolding to gain efficiency, while we strive to avoid that computation altogether.

Acknowledgements. We are grateful to Andrea Corradini and Eric Fabre for fruitful discussions on preliminary versions of this work.

References

1. J. Adamek, H. Herrlich, and G.E. Strecker. *Abstract and Concrete Categories - The Joy of Cats.* Wiley, 1990.
2. P. Baldan, S. Haar, and B. König. Distributed unfolding of petri nets. Technical Report CS-2006-1, Department of Computer Science, University Ca' Foscari of Venice, 2006.
3. A. Benveniste, E. Fabre, Claude Jard, and S. Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Trans. on Automatic Control*, 48(5):714–727, 2003.
4. A. Benveniste, S. Haar, E. Fabre, and C. Jard. Distributed monitoring of concurrent and asynchronous systems. In *Proc. of CONCUR'03*, volume 2761 of *LNCS*, pages 1–26. Springer, 2003.
5. R. Boel and J. van Schuppen. Decentralized failure diagnosis for discrete event systems with costly communication between diagnosers. In *Proc. 6th Int. Workshop on Discrete event Systems (WODES)*, pages 175–181, 2002.
6. C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic, 1999.

7. E. Fabre. Factorization of unfoldings for distributed tile systems, part 1: Reduced interaction case. Technical Report 4829, INRIA, May 2003.
8. E. Fabre. Factorization of unfoldings for distributed tile systems, part 2: General case. Technical Report 5186, INRIA, May 2004.
9. E. Fabre, A. Benveniste, S. Haar, and C. Jard. Distributed monitoring of concurrent and asynchronous systems. *Discrete Event Dynamic Systems: theory and application*, 15(1):33–84, 2005.
10. S. Genc and S. Lafortune. Distributed Diagnosis of discrete-event systems using Petri net unfoldings. In W.M.P. van der Aalst and E. Best, editors, *Proc. of ICATPN 2003*, volume 2679 of *LNCS*, pages 316–336. Springer, 2003.
11. K. Heljanko, V. Khomenko, and M. Koutny. Parallelisation of the petri net unfolding algorithm. In *Proc. of TACAS'02*, volume 2280 of *LNCS*, pages 371–385. Springer, 2002.
12. S. Mac Lane. *Categories for the working mathematician*. Springer, 1971.
13. J. Meseguer, U. Montanari, and V. Sassone. Process versus unfolding semantics for Place/Transition Petri nets. *Theoret. Comp. Sci.*, 153(1-2):171–210, 1996.
14. M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoret. Comp. Sci.*, 13:85–108, 1981.
15. W. Reisig. *Petri Nets. An Introduction*. Number 4 in EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1982.
16. S. L. Ricker and J. van Schuppen. Decentralized failure diagnosis with asynchronous communication between diagnosers,. In *Proc. of the European Control Conference*, 2001.
17. S.L. Ricker and K. Rudie. Distributed knowledge for communication in decentralized discrete-event systems. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, 2001.
18. G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, volume 1. World Scientific, 1997.
19. M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. on Automatic Control*, 40(9):1555–1575, 1995.
20. G. Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.