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
baldan@dsi.unive.it stefan.haar@irisa.fr koenigba@fmi.uni-stuttgart.de

Abstract. Petri net unfoldings have been recognised as an efficient means to fight state space explosion when dealing with concurrent and distributed systems. 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 between 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 this field, 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 [13] is a branching partial order semantics which represents in a single structure all the possible events in computations, and their mutual relationships: 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,

* 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), CRUI/DAAD Vigoni "Models based on Graph Transformation Systems: Analysis and Verification".

but also keeping together different branches of computations in its branching structure as long as possible, i.e., 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 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 between the various 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 observation. This deduction—or reconstruction—of faulty behaviours from the observations is the topic of diagnosis; efforts to force the system into a desired behaviour or to steer it away from an undesired one are studied in the domain of *control*. Although it has an important intersection with diagnosis (in terms of models and techniques), 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 [18]; 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 [5] for an overview and [15, 4] for details. Some authors consider timed extensions [16] or use Petri nets as system models [9].

The diagnosis approach in [2, 3, 8], 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 assumed to be 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 [2]). 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 lead to "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 the interactions with the other components. In [3, 8] Petri net components were fused on places, and the fusion of views was done through products of event structures obtained using a projection operation with an exchange of messages relating transition actions.

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 different components communicate through the interface net, whose unfolding is used to record 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. Interleaving structures have also a partial order representation which is proposed for efficiency reasons.

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. In §7 we introduce a partial order representation of interleaving structures. Finally, in §8 we draw some conclusions, with a look on related and future work.

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.

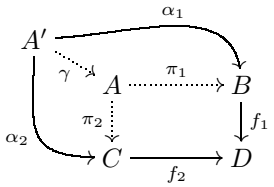
Let A be a set. The powerset of A is denoted by $\mathbf{2}^A$. A *multiset* of A is a total function $M\colon A \to \mathbb{N}$. It is called *finite* if the underlying set $\{a \in A \mid$

$M(a) > 0\}$ 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. Union and difference of multisets are denoted by $\oplus$ and $\ominus$, respectively. 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. The composition of two finitary multirelations $f : A \leftrightarrow B$ and $g : B \leftrightarrow C$ is the (finitary) multirelation $g \circ f : A \leftrightarrow C$ defined as $(g \circ f)(a, c) = \sum_{b\in B} f(a, b)\, g(b, c)$. 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 $a \in A$ we have $\sum_{b\in B} f(a, b) \geq 1$, *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]). Below we recall pullbacks and factorisation structures, of which we will make an extensive use.

Definition 1 (pullback). *Let* $\mathbf{C}$ *be a category, and* $f_1 : B \to D$, $f_2 : C \to D$ *arrows in* $\mathbf{C}$. *The* pullback *of* f_1 *and* f_2 *is an object* A *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* $\pi_i \circ \gamma = \alpha_i$ *(*$i \in \{1, 2\}$*).*



The object A *is called pullback object and denoted by* $B \times_D C$.

For instance, for a fixed 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*

– **C** *has* (E, M)-factorisations *of morphisms, i.e., each morphism* f *of* **C** *has a factorisation* $f = m \circ e$ *with* $e \in E$ *and* $m \in M$*.*
– **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. In the rest of the paper Λ denotes a fixed label set for all considered Petri nets. In order to obtain a category we are using Petri net morphisms as introduced in [19]. Note specifically that these morphisms preserve behaviour, i.e., "more behaviour" is allowed in the target net than in the source net.

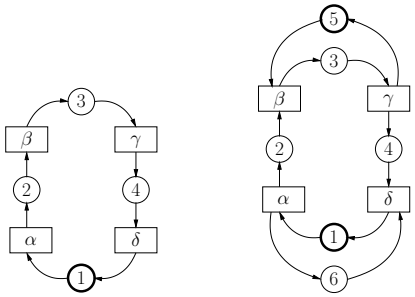
Definition 3 (Petri net). *A* Petri net *is a tuple* $N = (S, T, \lambda, {}^\bullet(), ()^\bullet, m)$ *where* S *is the set of places,* T *is the set of transitions,* $\lambda\colon T \to \Lambda$ *is a labelling function,* ${}^\bullet(), ()^\bullet\colon T \to \mathbf{2}^S$ *associate to each transition*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

A (Petri net) morphism $\tau = (\eta, \beta)\colon N \to N'$ *is a pair consisting of a partial function* $\eta\colon T \dashrightarrow T'$ *and a finitary multirelation* $\beta\colon S \leftrightarrow S'$ *such that*

1. $\mu\beta(m) = m'$*;*
2. *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***.*

Note that in the above definition the initial marking, and pre- and post-sets of transitions are proper sets, rather than multisets. Hence the category **Safe**



(a) N_3' (b) N_3

Fig. 1. Two examples of Petri nets.

in [19] is a full subcategory of **PN**, which in turn is a full subcategory of the category of semi-weighted nets in [12]. This choice will simplify the presentation; it can be seen that the results in the mentioned papers which are relevant to our setting (notably Proposition 10) trivially adapt to our case.

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 by 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 Petri net 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.*

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:*

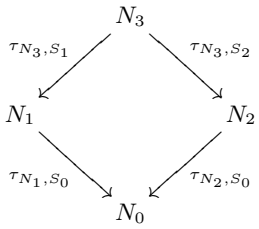


Fig. 2. Decomposition of Petri nets.

- $T_0 = \{(t,0) \mid t \in T \wedge ({}^\bullet t \cap S_0 \neq \emptyset \ \vee \ t^\bullet \cap S_0 \neq \emptyset)\}$,
- $\lambda_0((t,0)) = \lambda(t)$,
- ${}^\bullet(t,0) = {}^\bullet t \cap S_0$, $(t,0)^\bullet = t^\bullet \cap S_0$ *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.*

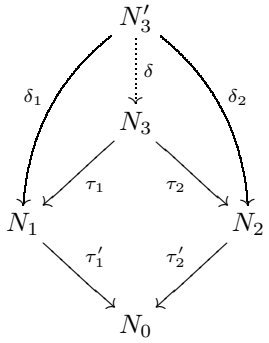
The next proposition provides a recipe for decomposing Petri nets along some chosen places, which play the role of interface between the subcomponents.

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:* $N_1 = [N_3]_{S_1}$, $N_2 = [N_3]_{S_2}$ *and* $N_0 = [N_1]_{S_0} = [N_2]_{S_0}$ *(see Fig. 2). 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 (formally for any* $t_1 \in T_1$, $t_2 \in T_2$ *if* $\lambda_1(t_1) = \lambda_2(t_2)$ *then* $t_1 \in T_0$ *or* $t_2 \in T_0$*). Then the diagram in Fig. 2 is a pullback diagram.*

Proof. As abbreviation we set $\tau_i = (\eta_i, \beta_i) = \tau_{N_3,S_i}$, and $\tau_i' = (\eta_i', \beta_i') = \tau_{N_i,S_0}$ for $i \in \{1,2\}$.

Assume that there is a net N_3' with morphisms $\delta_i = (\eta_{\delta_i}, \beta_{\delta_i})\colon N_3 \to N_i$, $i \in \{1,2\}$ such that $\tau_1' \circ \delta_1 = \tau_2' \circ \delta_2$ (see diagram below). Our aim is to show the existence of a unique morphism $\delta = (\eta, \beta)\colon N_3' \to N_3$ such that $\tau_i \circ \delta = \delta_i$ for

$i \in \{1, 2\}$.



First observe that T_3 is (isomorphic to) the pullback of η_1' and η_2' in the category $\mathbf{LSet}^*$ (where the labelling is given by the labelling function of N_3). This follows from the fact that every transition in T_3 has a non-empty pre-set and therefore has an image in at least one component net and from the assumption on the labelling of local transition. Hence there is a unique mapping $\eta\colon T_3' \dashrightarrow T_3$ such that $\eta_i \circ \eta = \eta_{\delta_i}$ for $i \in \{1, 2\}$.

In order to obtain β we show that S_3 is the pullback of β_1' and β_2' in the category of multi-relations. We know that $\beta_1' \circ \beta_{\delta_1} = \beta_2' \circ \beta_{\delta_2}$. We assume that an element $s' \in S_3'$ has the following images

$$\mu\beta_{\delta_1}(s') = \bigoplus_{s_1 \in S_1} k_{s_1}^{s'} \cdot s_1 \qquad \mu\beta_{\delta_2}(s) = \bigoplus_{s_2 \in S_2} \ell_{s_2}^{s'} \cdot s_2.$$

It can be shown that whenever $\beta_1'(s_1) = \beta_2'(s_2)$, then $k_{s_1}^{s'} = \ell_{s_2}^{s'}$. This follows directly from the commutativity of the diagram and the fact that β_1' and β_2' are injective (we even have that $\beta_i'(s_i) = s_i$ when defined). Now define a mediating multi-relation $\beta\colon S_3' \leftrightarrow S_3$ as follows:

$$\mu\beta(s') = \bigoplus_{s_1 \in S_0} k_{s_1}^{s'} \cdot s_1 \oplus \bigoplus_{s_1 \in S_1 \setminus S_0} k_{s_1}^{s'} \cdot s_1 \oplus \bigoplus_{s_2 \in S_2 \setminus S_0} \ell_{s_2}^{s'} \cdot s_2.$$

It can be easily seen that $\beta_1 \circ \beta = \beta_{\delta_1}$ and $\beta_1 \circ \beta = \beta_{\delta_2}$. Furthermore since β_1, β_2 are injective, β is the only multi-relation that can make the diagram commute.

The fact that η is a mediating arrow in $\mathbf{LSet}^*$ and β is a mediating arrow in the category of multirelations allow to conclude uniqueness and commutativity for $\delta = (\eta, \beta)$.

It is left to show that $\delta = (\eta, \beta)$ is a Petri net morphism. Specifically we will in the following show that $\mu\beta({}^\bullet t') = {}^\bullet\eta(t')$ for a transition t' of N_3' (for the remaining conditions the proof is analogous). We distinguish the following cases:

- $\eta_{\delta_1}(t') \downarrow$ and $\eta_{\delta_2}(t') \downarrow$.
  Let $t_1 = \eta_{\delta_1}(t')$, $t_2 = \eta_{\delta_2}(t')$ and $t = \eta(t')$. Note that since local transitions in different nets must have different labels and t_1, t_2 have the same label, there exists necessarily a transition t_0 in N_0 such that $\eta_1'(t_1) = t_0 = \eta_2'(t_2)$.

From the decomposition construction we know that ${}^\bullet t = {}^\bullet t_1 \oplus {}^\bullet t_2 \ominus {}^\bullet t_0$. (Remember that places have the same names in all the nets N_0, N_1, N_2, N_3.) Furthermore, according to the definition of β, it holds that

$$\begin{aligned}\mu\beta({}^\bullet t') &= \bigoplus_{s' \in {}^\bullet t'} \mu\beta(s') \\ &= \bigoplus_{s' \in {}^\bullet t'} \left(\bigoplus_{s_1 \in S_0} k_{s_1}^{s'} \cdot s_1 \oplus \bigoplus_{s_1 \in S_1 \setminus S_0} k_{s_1}^{s'} \cdot s_1 \oplus \bigoplus_{s_2 \in S_2 \setminus S_0} \ell_{s_2}^{s'} \cdot s_2 \right)\end{aligned}$$

From the definition of β_{δ_1} we conclude that

$${}^\bullet t_1 = \mu\beta_{\delta_1}({}^\bullet t') = \bigoplus_{s' \in {}^\bullet t'} \bigoplus_{s_1 \in S_1} k_{s_1}^{s'} \cdot s_1.$$

An analogous equation can be derived for ${}^\bullet t_2$. Finally we can represent ${}^\bullet t_0$ as follows:

$${}^\bullet t_0 = {}^\bullet(\eta_1' \circ \eta_{\delta_1})(t') = \mu(\beta_1' \circ \beta_{\delta_1})({}^\bullet t') = \bigoplus_{s' \in {}^\bullet t'} \bigoplus_{s_1 \in S_0} k_{s_1}^{s'} \cdot s_1.$$

Summing everything up (and remembering that $k_{s_1}^{s'} = \ell_{s_2}^{s'}$ for $s_1 = s_2 \in S_0$) we obtain:

$$\mu\beta({}^\bullet t') = {}^\bullet t_1 \oplus {}^\bullet t_2 \ominus {}^\bullet t_0 = {}^\bullet t = {}^\bullet \eta(t').$$

- $\eta_{\delta_1}(t') \downarrow$ and $\eta_{\delta_2}(t') \uparrow$.
  We set $t_1 = \eta_{\delta_1}(t')$ and necessarily $\eta_1'(t_1)$ must be undefined. This implies that $t = \eta(t')$ is *only* connected to places in S_1, and not to places in S_2. Hence we have ${}^\bullet t = {}^\bullet t_1$. Furthermore we have ${}^\bullet t' = m_1 \oplus m$, where m_1 contains all places mapped to a place of N_1 and m contains all places not mapped anywhere (as above). Because of $\mu\beta_{\delta_2}({}^\bullet t') = {}^\bullet \eta_{\delta_2}(t') = \emptyset$ there can be no places of any other kind.
  It can be shown that $\mu\beta(m_1) = {}^\bullet t_1$ and $\mu\beta(m) = \emptyset$. Hence we can conclude that

  $$\mu\beta({}^\bullet t') = \mu\beta(m_1 \oplus m) = \mu\beta(m_1) \oplus \mu\beta(m) = {}^\bullet t_1 \oplus \emptyset = {}^\bullet t = {}^\bullet \eta(t').$$

- $\eta_{\delta_1}(t') \uparrow$ and $\eta_{\delta_2}(t') \downarrow$.
  Analogous to the case above.
- $\eta_{\delta_1}(t') \uparrow$ and $\eta_{\delta_2}(t') \uparrow$.
  Since any transition in N_3 appears either in N_1 or in N_2 (or in both), necessarily $\eta(t')$ is undefined. Reasoning as in earlier cases we deduce that $\mu\beta({}^\bullet t') = \emptyset$, as desired.

□

Note that, in order to exploit the results about the unfolding, also the component nets must not contain transitions with empty pre-set. For safe nets this

can be achieved by introducing extra complement places. Henceforth, all decompositions are supposed to have this property.

Additionally, decomposition will have to be done in such a way that local transitions in different components have different labels. Note that, for instance, in our application area, i.e., the area of diagnosis, it is not restrictive to assume that transitions in different components can be distinguished and thus are assigned different labels. It is worth remarking that, from a technical point of view, such requirement implies, in particular, that if a transition t of N_3 occurs both in N_1 and N_2 then it must occur also in N_0.

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 decomposition 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 [14]) 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. 3. Note that neither N_1 nor N_2 are safe (or even bounded).

4 Unfolding Petri Nets

In this section we recap the unfolding semantics for Petri nets and we discuss some of its properties which will play a basic role in the development.

Recall that given a Petri net N the dependencies between transitions are captured by two basic relations, causality and conflict.

Definition 7 (causality, conflict). *Let N be a Petri net.* 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 $x \in S \cup T$, $\lfloor x \rfloor = \{y \in S \cup T \mid y \leq_N x\}$.*

Conflict *is the least 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 8 (occurrence net). *An* occurrence net *is a net N satisfying the following requirements:*

1. *if $t^\bullet \cap t'^\bullet \neq \emptyset$ then $t = t'$ (any place is in the post-set of at most one transition);*
2. *$\leq_N$ is a partial order and $\lfloor t \rfloor$ is finite for any $t \in T$;*

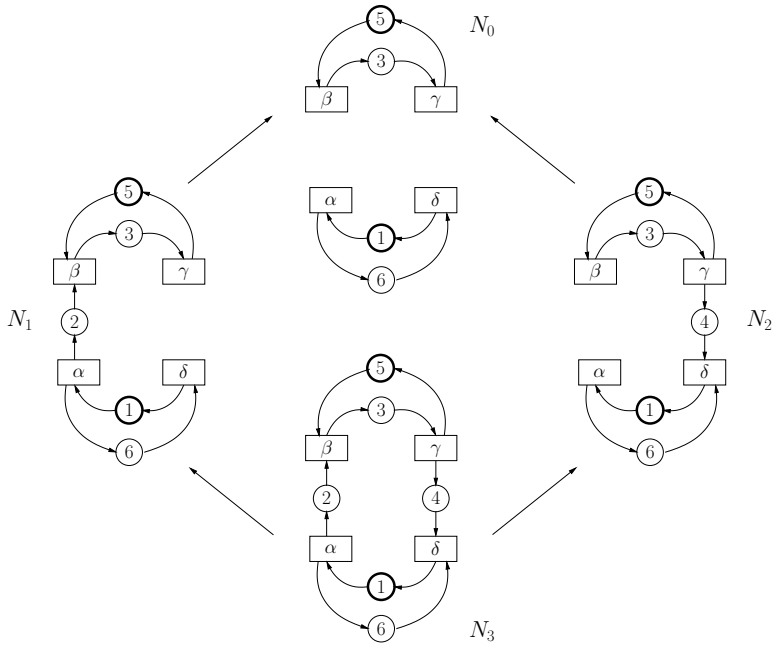


Fig. 3. Decomposing a loop as a pullback of nets.

3. *the* initial marking m *is the set of* $\leq_N$*-minimal places;*
4. $\#_N$ *is irreflexive.*

We denote by **ON** *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 9 (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:*

$$m' = \{\langle \emptyset, s \rangle \mid s \in m\}$$
$$S' = m' \cup \{\langle \{t'\}, s \rangle \mid t' = \langle X, t \rangle \in T' \ \wedge \ s \in t^\bullet\}$$
$$T' = \{\langle X, t \rangle \mid X \subseteq S' \ \wedge \ conc(X) \ \wedge \ t \in T \ \wedge \ \mu\beta(X) = {}^\bullet t\}$$

$$\textit{For } t' = \langle X, t\rangle \in T' : \qquad {}^\bullet t' = X \quad \textit{and} \quad t'^\bullet = \{\langle \{t'\}, s\rangle \mid s \in t^\bullet\}$$

$$\eta(t') = t \quad \text{iff} \quad t' = \langle X, t\rangle$$
$$\beta(s', s) = 1 \quad \text{iff} \quad s' = \langle x, s\rangle \quad (x \in \mathbf{2}^{T'})$$

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 Petri net; transition $t' = \langle X, t\rangle$ represents a firing of t that consumes the resources in X.

Proposition 10 (right adjoint [19, 12]). *The construction $\mathcal{U}$ of Definition 9 extends to a functor $\mathcal{U} : \mathbf{PN} \to \mathbf{ON}$, which is right adjoint to the inclusion of* **ON** *into* **PN**.

Right adjoints preserve limits (see [11, 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. 3 we obtain the occurrence nets O_0, O_1, O_2 and O_3 in Fig. 4 (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 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 [19], Lemma 3.3.6).

Definition 11 (projection of occurrence nets). *Let $\tau = (\eta, \beta) : O_1 \to O_2$ be an* **ON** *morphism and let $\tau^p : O_1 \to O_2^1$, $\tau^e : O_2^1 \to O_2$ be the* pe*-factorisation of τ. Then the occurrence net O_2^1 is called the* projection *of O_1 onto O_2.*

Roughly in O_2^1 we leave out all elements (places and transitions) of O_2 that are not in the image of τ.

Example: Consider the unfoldings of our running example in Fig. 4. 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

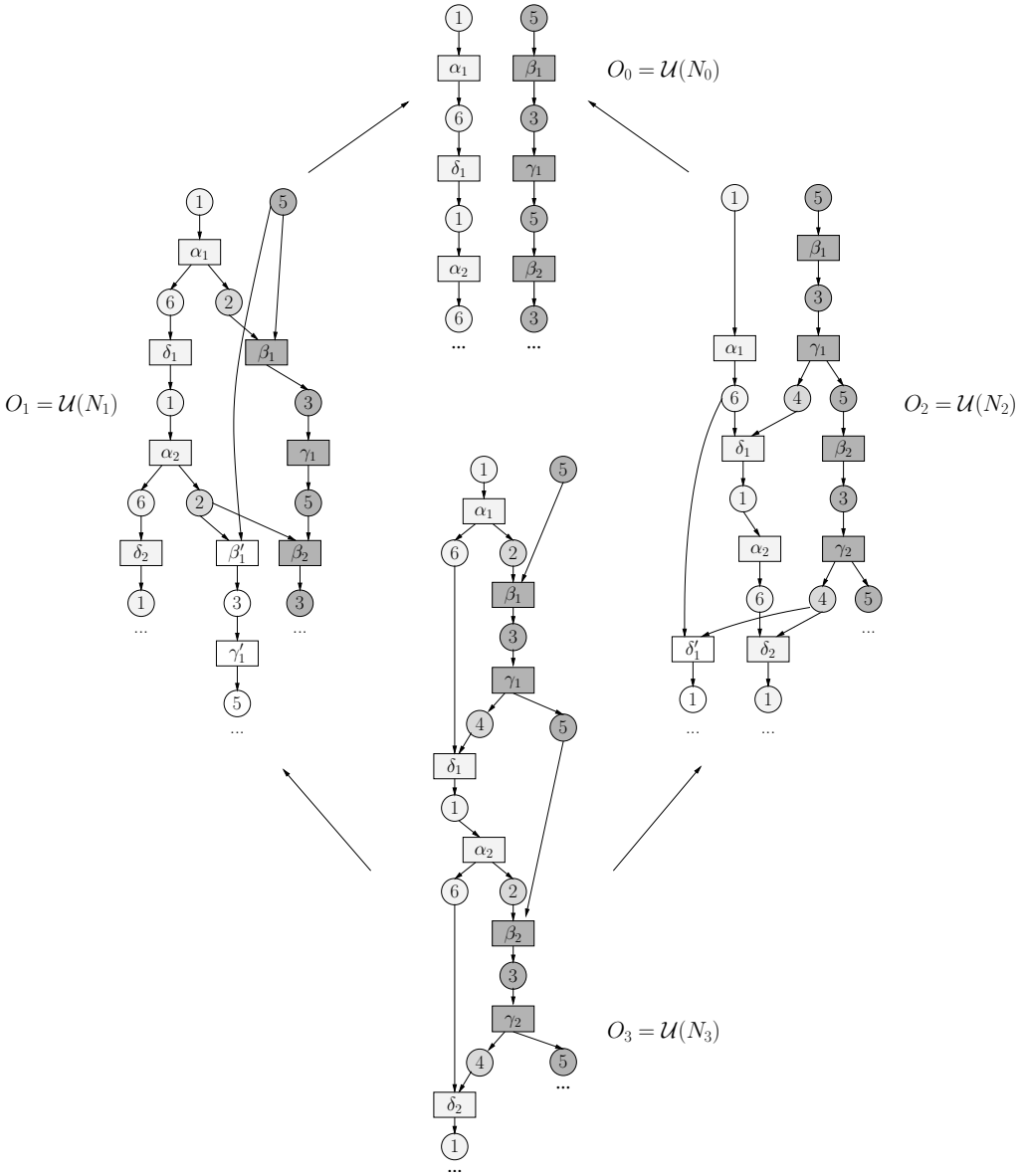


Fig. 4. Composition of unfoldings as pullback of occurrence nets.

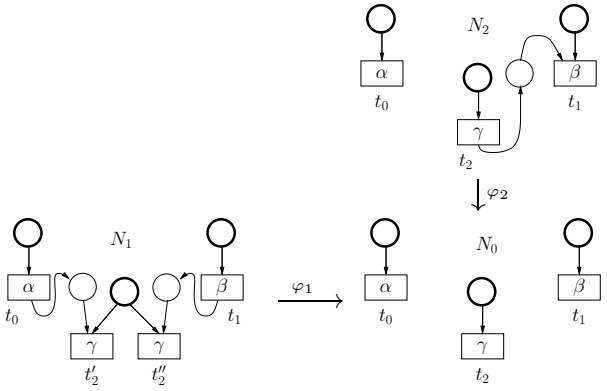


Fig. 5. Projecting dependency relations over the interface.

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 components. 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 component net does not work. Consider, for instance, the occurrence nets in Fig. 5, 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, i.e., in saying that t_2 can be caused by either t_0 or t_1, 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. However, as mentioned above, prime event structures are not sufficiently rich to model this situation. It can be shown that similar problems arise when considering more general partial order models including Winskel's general event structures [19].

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, which one would like to have for efficiency reasons. Hence our

solution is the following: in order to obtain a clean theory we work with interleavings for now and explain afterwards how they can be efficiently represented by a data structure based on partial orders and specifically on occurrence nets (see Section 7).

5.1 Interleaving structures

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 in the obvious way (apply f pointwise, removing from the sequence elements on which f is undefined.)

Definition 12 (interleaving structures). *An* interleaving structure *is a tuple* $\mathcal{I} = (E, R, \lambda)$ *where* E *is a set of* events, $\lambda\colon E \to \Lambda$ *a labelling of events and* $R \subseteq E^{\odot}$, *called the set of* runs, *satisfying:*

1. R *is prefix-closed,*
2. R *contains the empty run* ε,
3. *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 13 (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*

1. $\lambda_2(\theta(e)) = \lambda_1(e)$ *whenever* $\theta(e) \downarrow$ *(i.e., function* θ *preserves labels)*
2. *for every* $r \in R_1$ *it holds that* $\theta(r) \in R_2$.

The category of interleaving structures and interleaving morphisms is denoted by **Ilv**.

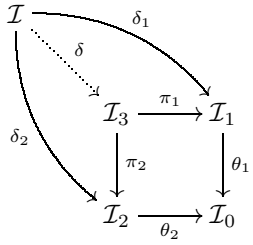
Observe that by Condition (2) above, θ must be injective on the events occurring in any single run. Below we show that pullbacks can be constructed in a quite straightforward way in **Ilv**.

Proposition 14 (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.

Proof. Let $\mathcal{I}$ be another interleaving structure with morphisms $\delta_i : \mathcal{I} \to \mathcal{I}_1$ such that $\theta_1 \circ \delta_1 = \theta_2 \circ \delta_2$. We have to show that there exists a unique morphism $\delta : \mathcal{I} \to \mathcal{I}_3$ such that $\pi_i \circ \delta = \delta_i$. This is illustrated by the diagram below.



Since E'_3 is the pullback of θ_1, θ_2 in the category $\mathbf{LSet}^*$ of labelled sets and partial functions we obtain a unique function $\delta : E \dashrightarrow E'_3$ such that the diagram commutes in $\mathbf{LSet}^*$.

Let us first observe that for any run $r \in R$ we have $\delta(r) \in R_3$. In fact, $\delta_i(r) \in R_i$ for $i \in \{1, 2\}$. Thus, since $\pi'_i(\delta(r)) = \delta_i(r)$ for $i \in \{1, 2\}$, we conclude that $\delta(r) \in R_3$. This immediately implies that $\delta(E) \subseteq E_3$, and thus we can view δ as a partial function $E \dashrightarrow E_3$. In fact, given any $e \in E$ there must be a run r where e occurs. Since $\delta(r) \in R_3$, we deduce that either $\delta(e) \uparrow$ or $\delta(e) \in E_3$.

It remains to show that δ is an interleaving morphism. We have already proved that for any $r \in R$ its image $\delta(r) \in R_3$. Furthermore for an event $e \in E$ it holds that $\lambda(\delta(e)) = \lambda(\pi_i(\delta(e))) = \lambda(\delta_i(e)) = \lambda(e)$ whenever $\pi_i(\delta(e))$ is defined for some i. If neither $\pi_1(\delta(e))$ nor $\pi_2(\delta(e))$ is defined, then $\delta(e)$ must be undefined. Hence we conclude. □

5.2 Factorisation Structures

We next study factorisations of interleaving structures, showing 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 15 (projection, embedding). *An interleaving morphism $\theta : \mathcal{I}_1 \to \mathcal{I}_2$ is called* projection *if the induced function on runs $\theta : R_1 \to R_2$ is surjective. Morphism θ is called* embedding *if the mapping on events is a total injection.*

Observe that by definition any projection $\theta : \mathcal{I}_1 \to \mathcal{I}_2$ is surjective on the set of events. In fact, any event $e_2 \in E_2$ occurs in at least one run $r_2 \in R_2$. Since θ is a projection, there exists $r_1 \in R_1$ such that $\theta(r_1) = r_2$ and thus there must be some event $e_1 \in E_1$, occurring in r_1, such that $\theta(e_1) = e_2$.

Given any morphism $\theta : \mathcal{I}_1 \to \mathcal{I}_2$ 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.

Next we show that the category of interleaving structures is *pe*-structured.

Proposition 16 (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.*

Proof. The fact that E and M are closed under isomorphisms is obvious, while the possibility of factorising any morphism follows from the considerations in Section 5.

It remains to show that **Ilv** has the unique (E, M)-diagonalisation property. Consider a square as below where $e \in E$, $m \in M$

$$\begin{array}{ccc} I_A & \xrightarrow{e} & I_B \\ {\scriptstyle f}\downarrow & \overset{d}{\swarrow} & \downarrow{\scriptstyle g} \\ I_C & \xrightarrow{m} & I_D \end{array}$$

and let us show the existence and uniqueness of a morphism d making the diagram commute. Morphism d exists uniquely in $\mathbf{Set}^*$ since e is epi and m is mono in $\mathbf{Set}^*$, and it can be defined by assigning to e_B the unique $e_C \in E_C$ such that $m(e_C) = g(e_B)$.

The function d is an **Ilv**-morphism. In fact, for any $r_B \in R_B$, let $r_B = e_1 \dots e_n$. Since e is a projection there exists $r_A \in R_A$ such that $e(r_A) = r_B$. Thus it is easy to conclude that $d(r_B) = f(r_A) \in R_C$, as desired. □

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

Proposition 17 (stability of projections under pullbacks). *In the category* **Ilv** *projections are stable under pullbacks.*

Proof. Let $\pi_i : \mathcal{I}_3 \to \mathcal{I}_i$, $\theta_i : \mathcal{I}_i \to \mathcal{I}_0$ for $i \in \{1, 2\}$ be a pullback in the category of interleaving structures and assume that θ_1 is a projection.

$$\begin{array}{ccc} \mathcal{I}_3 & \xrightarrow{\pi_1} & \mathcal{I}_1 \\ {\scriptstyle \pi_2}\downarrow & & \downarrow{\scriptstyle \theta_1} \\ \mathcal{I}_2 & \xrightarrow{\theta_2} & \mathcal{I}_0 \end{array}$$

We have to show that π_2 is a projection. Take a run $r_2 \in R_2$. We have $\theta_2(r_2) \in R_0$ and hence—since θ_1 is a projection—there exists a run $r_1 \in R_1$ with $\theta_1(r_1) = \theta_2(r_2)$. The situation is as follows: $r_0 = e_1 \dots e_n$ and we have $r_i = r_0^i e_1^i r_1^i \dots r_{n-1}^i e_n^i r_n^i$ where $\theta_i(e_j^i) = e_j$ and $\theta^e(r_j^i) = \varepsilon$ (the empty run).

We now set $s_j^1 = (e_1', *) \dots (e_k', *)$ whenever $r_j^1 = e_1' \dots e_k'$ and $s_j^2 = (*, e_1') \dots (*, e_k')$ whenever $r_j^2 = e_1' \dots e_k'$. We define

$$r_3 = s_0^1 s_0^2 (e_1^1, e_1^2) s_1^1 s_1^2 \dots s_{n-1}^1 s_{n-1}^2 (e_n^1, e_n^2) s_n^1 s_n^2$$

and it is easy to see that $\pi_i(r_3) = r_i$ for $i \in \{1, 2\}$. Hence $r_3 \in R_3$ by Proposition 14. Summing up, we have shown that for any $r_2 \in R_2$ there exists a run $r_3 \in R_3$ such that $\pi_2(r_3) = r_2$. Thus π_2 is a projection. □

5.3 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 18. *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.

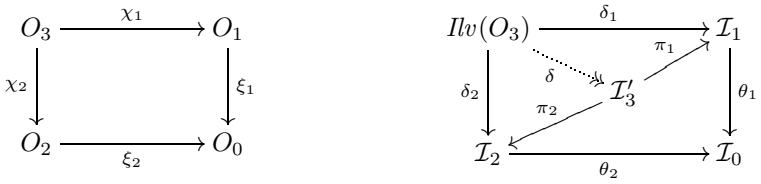
Lemma 19. *The mapping Ilv can be extended to a functor $Ilv : \mathbf{ON} \to \mathbf{Ilv}$.*

Proof. The only thing to show is that for any occurrence net morphism $\tau = (\eta, \beta) : O_1 \to O_2$ the mapping $\eta : Ilv(O_1) \to Ilv(O_2)$ is an **Ilv**-morphism, i.e., that for any $r_1 \in R_{Ilv(O_1)}$ its image $\eta(r_1) \in R_{Ilv(O_2)}$. This follows immediately from the fact that occurrence net morphisms preserve configurations (see, e.g., [19]. □

The functor Ilv does not preserve pullbacks. Consider for instance the pullback in **ON** depicted in Fig. 6. Transitions marked with α_i respectively β_i have label α and β and their indices are used to represent the **ON** morphisms. The corresponding interleaving structures $Ilv(O_i)$ for $i \in \{0, 1, 2\}$ has sets of runs $\{\varepsilon, \alpha\}$, $\{\varepsilon, \alpha, \alpha\beta\}$ and $\{\varepsilon, \alpha_1, \alpha_2\}$, respectively. If we take the pullback of $Ilv(\varphi_1)$ and $Ilv(\varphi_2)$ we obtain an interleaving structure with *three* (instead of four) events α_1, α_2, β. The set of runs is the prefix closure of $\{\alpha_1\beta,\ \alpha_2\beta\}$.

Still we can establish a useful relation between pullbacks in the category of occurrence nets and in **Ilv**.

Lemma 20. *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.*



Proof (Sketch). First notice that we can decompose the functor Ilv into the standard functor $Ev : \mathbf{ON} \to \mathbf{PES}$ from occurrence nets to the category **PES** of prime event structures [19], which is a right adjoint, and a functor $Ilv' : \mathbf{PES} \to$

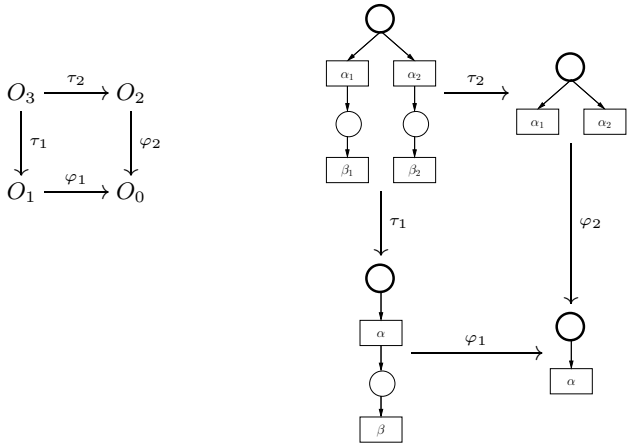
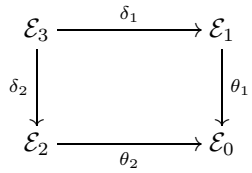


Fig. 6. Functor *Ilv* does not preserve pullbacks.

Ilv. Thus, if we take the prime event structures $\mathcal{E}_i$ underlying the occurrence nets O_i, i.e., $\mathcal{E}_i = Ev(O_i)$, we obtain a pullback square in the category **PES**, as illustrated below.



Thus we have to show that there is a projection from $Ilv'(\mathcal{E}_3)$ to $\mathcal{I}'_3$.

Let $r_3 = e_1 \dots e_n \in R'_3$ be a run of $\mathcal{I}'_3$. Hence there exist runs $r_i = \pi_i(r_3) \in R_i$ which correspond to configurations in $\mathcal{E}_1$, $\mathcal{E}_2$. Now let $\mathcal{E}$ be an event structure with events $E = \{e_1, \dots, e_n\}$ such that e_{i+1} causally depends on e_i. There exist morphisms $\delta'_i \colon \mathcal{E} \to \mathcal{E}_i$ such that $\delta'_i(e_j) = \pi_i(e_j)$, since $\pi_i(E)$ is a configuration in $\mathcal{E}_i$ that can be "executed" in that order.

Furthermore commutativity, i.e., $\theta_1 \circ \delta'_1 = \theta_2 \circ \delta'_2$ follows from the commutativity of the diagram consisting of $\theta_1, \theta_2, \pi_1, \pi_2$. Since $\mathcal{E}_3$ is the pullback of $\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_2$ there must be a unique morphism $\delta' \colon \mathcal{E} \to \mathcal{E}_3$ such that $\delta_i \circ \delta' = \delta'_i$.

Since Ilv' is a functor, we can also consider δ' as an interleaving morphism. Hence $r = \delta'(e_1) \dots \delta'(e_n)$ is a run in $Ilv(O_3)$. We have that $\pi_i(\delta(\delta'(e_j))) = \delta_i(\delta'(e_j)) = \delta'_i(e_j) = \pi_i(e_j)$. The elements of E_3 are uniquely determined by their projections and hence we have $\delta(\delta'(e_j)) = e_j$. This implies that $\delta(r) = r_3$ and thus δ is a projection as required. □

We can now relate projections of occurrence nets to projections of interleaving structures.

Lemma 21. *Let $\tau: O_1 \to O_2$ be an occurrence net morphism. The projection of O_1 to O_2 can be obtained by taking the interleaving structure morphism $\theta = Ilv(\tau): \mathcal{I}_1 \to \mathcal{I}_2$ and determining the projection-embedding factorisation $\mathcal{I}_1 \stackrel{\theta^p}{\to} \mathcal{I}_2^1 \stackrel{\theta^e}{\to} \mathcal{I}_2$ of θ. In order to obtain the projection O_2^1 take the subnet of O_2 containing only transitions in the image of θ^e.*

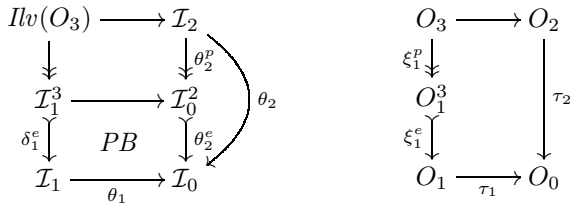
Proof. This is an easy consequence of the characterisation of the projection-embedding factorisations in the two categories **Ilv** and **ON** (respectively **PN**). □

Summing up, we obtain a procedure for determining the projection of a pullback object in the category of occurrence nets without actually constructing the pullback.

Proposition 22. *Let $\tau_i: O_i \to O_0$, $i \in \{1,2\}$ be two occurrence net morphisms and let $\xi_i: O_3 \to O_i$, $i \in \{1,2\}$ 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:*

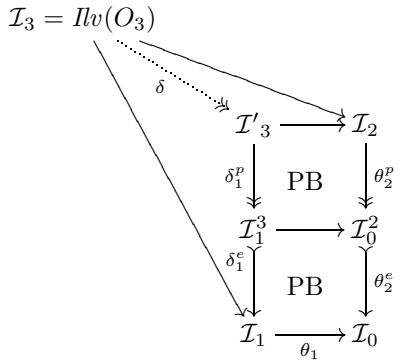
- *Determine the interleaving structures $\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2$ corresponding to O_0, O_1, O_2, i.e., $\mathcal{I}_i = Ilv(O_i)$, including their morphisms $\theta_i = Ilv(\tau_i): \mathcal{I}_i \to \mathcal{I}_0$, $i \in \{1,2\}$.*
- *Compute the projection-embedding factorisation $\mathcal{I}_2 \stackrel{\theta_2^p}{\to} \mathcal{I}_0^2 \stackrel{\theta_2^e}{\to} \mathcal{I}_0$ of θ_2.*
- *Take the pullback of θ_1 and θ_2^e and obtain the morphism $\delta_1^e: \mathcal{I}_1^3 \to \mathcal{I}_1$.*
- *Now take the subnet of O_1 that contains the transitions in the image of δ_1^e (as described in Lemma 21).*

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



Proof. We start by constructing $\mathcal{I}_3'$ as the pullback of θ_1 and θ_2. This gives $\delta: \mathcal{I}_3 \to \mathcal{I}_3'$ with $\mathcal{I}_3 = Ilv(O_3)$ as a mediating morphism, which according to Lemma 20 is a projection.

Then we take the pullback of θ_1 and θ_2^e resulting in a split of the larger pullback as shown below. Specifically, since θ_2^e is an embedding and thus it is preserved by pullbacks, we conclude that also δ_1^e is an embedding. In the same way Proposition 17 implies that δ_1^p is a projection.



Since projections compose, we can conclude that $\delta_1^p \circ \delta$ is also a projection, i.e., the pair $\delta_1^p \circ \delta$, δ_1^e is the unique (projection,embedding)-factorisation of $\mathit{Ilv}(\xi_1)$.

Hence, according to Lemma 21, we can restrict O_1 to the transitions in the image of δ_1^e and obtain O_1^3. □

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 described in Proposition 6. Then it builds, in a stepwise fashion, the remaining morphisms of the commuting diagram in Fig. 7, 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 $\eta_i(t)$.

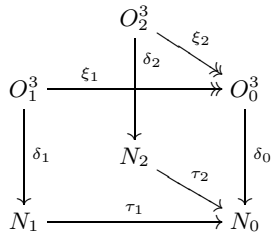


Fig. 7. Nets and morphisms involved in Algorithm 1.

Algorithm 1 (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 preset of a transition t in N_j and furthermore[4]
    (*) 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 postset $\{\langle\{t'_0\}, s_0\rangle \mid s_0 \in t_0{}^\bullet\}$ 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 $\langle\{t'\}, s\rangle \mapsto \langle\{t'_0\}, \tau_j(s)\rangle$.

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. Furthermore we will suggest in Section 7 how to efficiently check Condition (*).

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.

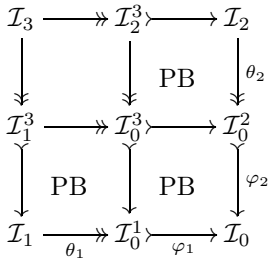
Example: The above algorithm, applied to our running example, produces the shaded subparts of the nets in Fig. 4. For instance transition β'_1 will never be added to $\bar{O}_1$. This transition may follow the run $\alpha_1\delta_1\alpha_2$, but there is no run r in O_2 for which we have $\xi_2(r) = \alpha_1\delta_1\alpha_2 = \xi_1(\alpha_1\delta_1\alpha_2)$.

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 23 (correctness of distributed unfolding). *We denote the (infinite) unions of the sequences of nets produced by the algorithm above by $\bar{O}_0$, $\bar{O}_1$ and $\bar{O}_2$. 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 $\mathcal{U}(N_i)$.*

[4] 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}.

Proof. Let $\mathcal{I}_i = \mathit{Ilv}(O_i)$ where $O_i = \mathcal{U}(N_i)$ for $i \in \{0,1,2\}$. Consider the diagram below which is obtained by determining the (projection,embedding)-factorisations in **Ilv** and where all squares—apart from the top left square—are pullbacks. The pullback consisting of the composition of the two bottom squares is determined as explained in Proposition 22. Similar considerations apply to the pullback consiting of the composition of the two right squares. Then—by standard pullback decomposition—we can split these two pullbacks obtaining the bottom right square. From the fact that projections and embeddings are preserved by pullbacks, we can then classify the arrows accordingly as shown in the diagram below.



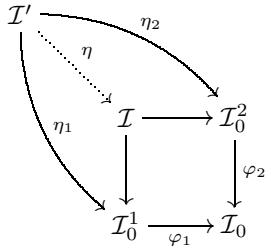
We will assume that all embeddings preserve the identities of items, i.e., that they map any transition t to itself. Since this fixes the mappings φ_i, it follows that $\theta_i(t) = \bar{\xi}_i(t)$ when applied to a transition t of $\bar{O}_i$.

In the following we will denote the restrictions of the occurrence nets $\bar{O}_0$, $\bar{O}_1$, $\bar{O}_2$ to valid transitions by $\hat{O}_0$, $\hat{O}_1$, $\hat{O}_2$. We start by showing that $\hat{O}_0$ equals O_0^3. Clearly $\hat{O}_0$ is a subnet of O_0 and, but construction, it contains exactly the transitions occurring in the runs of $\bar{\xi}_1(R_{\mathit{Ilv}(\bar{O}_1)}) \cap \bar{\xi}_2(R_{\mathit{Ilv}(\bar{O}_2)})$. Let us denote the interleaving structure with all these runs and all events that are contained in at least one run by $\mathcal{I}$.

According to Lemma 21 it is sufficient to prove that $\mathcal{I}$ is equal to $\mathcal{I}_0^3$. We will do this by showing that $\mathcal{I}$ is the pullback object of $\mathcal{I}_0^1 \to \mathcal{I}_0$ and $\mathcal{I}_0^2 \to \mathcal{I}_0$.

For this we first have to show that there are embeddings $\mathcal{I} \to \mathcal{I}_0^i$ for $i \in \{1,2\}$. We actually prove that $R_{\mathcal{I}} \subseteq R_{\mathcal{I}_0^i}$. Let r be a run of $\mathcal{I}$. It follows that $r \in \bar{\xi}_1(R_{\mathit{Ilv}(\bar{O}_1)})$, i.e., there exists a run r' of $\bar{O}_1$ with $\bar{\xi}_1(r') = r$. This implies that $r = \bar{\xi}_1(r') = \theta_1(r')$ and thus, by the properties of interleaving morphisms, $r \in R_{\mathcal{I}_0^1}$. The condition for $\mathcal{I}_0^2$ can be shown analogously.

Now, assume that there is an interleaving structure $\mathcal{I}'$ with morphisms $\eta_i \colon \mathcal{I}' \to \mathcal{I}_0^i$ for $i \in \{1,2\}$ such that $\varphi_1 \circ \eta_1 = \varphi_2 \circ \eta_2$. We will show that there exists a unique morphism $\eta \colon \mathcal{I}' \to \mathcal{I}$ with $\eta_i(r') = \eta(r')$ for every run r' of $\mathcal{I}'$ (see diagram below).



The only way to define η is to set $\eta(t) = \eta_1(t) = \eta_2(t)$. Note that $\eta_1(t)$ is defined if and only if $\eta_2(t)$ is defined and both elements are equal (this follows from the fact that $\varphi_i(t) = t$). This implies immediately that $\eta_i(r') = \eta(r')$ for every run r' of $\mathcal{I}'$. However it is not yet clear that $\eta: \mathcal{I}' \to \mathcal{I}$ is a well-defined interleaving structure morphism.

Hence we show that $\eta(r') \in R_{\mathcal{I}}$ for every run r' of $\mathcal{I}'$ by induction on the length of r'. If $r' = \varepsilon$, then we have $\eta(r') = \varepsilon$ and by definition $\varepsilon \in R_{\mathcal{I}}$. If $r' = r''t$, then we have, by the induction hypothesis, $\eta(r'') \in R_{\mathcal{I}}$.

Now we distinguish two cases: whenever $\eta_1(t)$ and $\eta_2(t)$ are undefined, we have that $\eta(t)$ is undefined, which implies that $\eta(r') = \eta(r'') \in R_{\mathcal{I}}$. If, on the other hand, $\eta_1(t) = \eta_2(t) = t_0$ for some transition t_0 of O_0, we conclude that—since $\eta_i(r''t)$ is a run of $\mathcal{I}_0^i$ and the θ_i are projections—there must be runs $r_i t_i$ of $\mathcal{I}_i$ with $\theta_i(r_i t_i) = \eta_i(r''t)$.

We can easily show by induction on the length of r_i that r_i must (eventually) be a run of $\bar{O}_i$. (Observe that for every prefix r_i' of r_i we have that $\bar{\xi}_i(r_i') = \theta_i(r_i')$ and $\theta_i(r_i')$ is—because of prefix closure—a run of $\mathcal{I}$ and hence of $\bar{\xi}_{3-i}(R_{Ilv(\bar{O}_{3-i})})$ and so Condition (*) of the algorithm is satisfied.)

Then—again by Condition (*)—the transition t_i can at some point be added to $\bar{O}_i$ and it will be since Condition (*) continues to hold. That means that at some point in the algorithm $r_i t_i$ will be contained in $Ilv(\bar{O}_i)$ and so $\eta(r') = \theta_i(r_i t_i) = \bar{\xi}_i(r_i t_i)$ will be contained in $\bar{\xi}_i(R_{Ilv(\bar{O}_i)})$ for $i \in \{1, 2\}$. This implies that $\eta(r')$ is contained in $\mathcal{I}$.

We continue by showing that $\hat{O}_1$ equals O_1^3. Again it is sufficient to show that $\hat{O}_1$ contains exactly the transitions contained in $\mathcal{I}_1^3$. We first show that every transition t of $\hat{O}_1$ is contained in $\mathcal{I}_1^3$. Since t is valid there exists a run rt in $\bar{O}_1$ such that $\bar{\xi}_1(rt) = \theta_1(rt) \in R_{\mathcal{I}} = R_{\mathcal{I}_0^3}$. Furthermore rt is also a run of O_1 and hence of $\mathcal{I}_1$. This means that rt is contained in $\mathcal{I}_1^3$ by pullback properties. And this implies that t is an event of $\mathcal{I}_1^3$.

For the other direction, let t be an event of $\mathcal{I}_1^3$. Hence there must be a run of the form rt in $\mathcal{I}_1^3$. If the image of t is defined, i.e., $\theta_1(t) = t'$, then there exists a run $\theta_1(rt) = r't'$ in $\mathcal{I}_0^3 = \mathcal{I}$. This means that all events of r can be added to $\bar{O}_1$ since they are all valid and their causes are also valid (see also Condition (*)). Finally t can be added to $\bar{O}_1$ as a valid transition. This implies that t is contained in $\hat{O}_1$. If instead $\theta_1(t)$ is undefined then we can show that it can be added to $\bar{O}_1$

using the same argument as above. Furthermore it is trivially valid and hence again contained in $\hat{O}_1$.

The case of $\hat{O}_2$ can be shown analogously. □

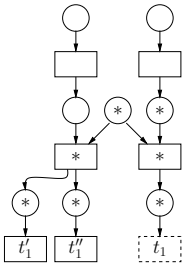
7 Partial Order Representation for Interleaving Structures

In order to obtain efficient data structures for storing interleaving structures, we represent an interleaving structure $\mathcal{I}$ (over events of a fixed occurrence net O) as an occurrence net morphism $\gamma\colon P \to O$, total but not necessarily injective, such that $\mathcal{I}$ results as the projection of $\mathit{Ilv}(P)$ along γ.

Recall that the distributed algorithm constructs the occurrence nets $\bar{O}_0$, $\bar{O}_1$, $\bar{O}_2$ with morphism $\xi_i : \bar{O}_i \to \bar{O}_0$ ($i \in \{1,2\}$). The interface manager needs to store the interleaving structures arising as the projection of $\mathit{Ilv}(\bar{O}_i)$ (for $i \in \{1,2\}$) over $\bar{O}_0$, and to operate on them. As suggested above, these interleaving structures are represented as morphisms $\gamma_i : P_i \to \bar{O}_0$, where the P_i are suitable occurrence nets.

We remark that the occurrence nets P_i and the morphisms γ_i can be constructed incrementally, without ever generating the represented interleaving structures. In fact, all the operations needed for the algorithm can be performed directly on this representation:

- *Adding a transition t_i to $\bar{O}_i$.*
  Assume, for instance, that a transition t_1 is added to the net $\bar{O}_1$. If $\xi_1(t_1) \uparrow$ then $\gamma_1 : P_1 \to O_1$ stays unchanged.
  If instead, $\xi_1(t_1) = t_0$, for some t_0 in $\bar{O}_0$, then we insert t_1 into P_1, but we also have to reconstruct the missing conflicts and causalities by using additional places. More specifically:
  - *Causality*: Take any place $s_1 \in {}^\bullet t_1$ on which ξ_1 is undefined. Then in $\bar{O}_1$ go backward in the causality chains that lead to s_1, until a transition t_1' in the domain of ξ_1 is reached in each chain. If T_1' is the set of all such transitions, add places to P_1 establishing a causal dependency between each maximal transition in T_1' and t_1.
  - *Conflict*: Take any place $s_1 \in {}^\bullet t_1$ on which ξ_1 is undefined. Then in $\bar{O}_1$ go backward in all causality chains that lead to s_1, until a transition in the domain of ξ_1 is reached in each chain. For any direct conflict with a transition t_1' found in this way, follow its causal descendants until a transition in the domain of ξ_1 is reached. If T_1' is the set of all such transitions, add places to P_1 establishing a conflict between each minimal transition in T_1' and t_1.
    For instance, assume that the net $\bar{O}_1$ to which we are adding transition t_1 has the shape depicted below and assume that morphism ξ_1 is undefined on items marked by $*$. Then we must add to P_1 places which induce the direct conflicts $t_1 \# t_1'$ and $t_1 \# t_1''$.



– *Intersection*
  The intersection of the set of runs of two interleaving structures represented by $\gamma_i \colon P_i \to O$ for $i \in \{1, 2\}$ can be computed by constructing the pullback of these morphisms in **ON**.

– *Existence of a run.*
  Given $\gamma_i \colon P_i \to \bar{O}_0$, a morphism $\xi_j \colon \bar{O}_j \to \bar{O}_0$ and a set of places X in $\bar{O}_j$, we want to check whether there exists a run r of $\bar{O}_j$ that contains all causes of X and none of its consequences with $\xi_j(r) \in R_{\gamma_i(Ilv(P_i))}$ (see Condition (*) in Algorithm 1).
  To this aim, add a dummy transition t' with ${}^\bullet t' = X$ to $\bar{O}_j$ and set $\xi_j(t') \uparrow$. Then take the pullback of (the new) ξ_j and γ_i and check whether the pullback net contains a transition that is mapped to t'.
  Similarly this procedure can be used to check the validity of transitions.

We remark that that—although not detailed in this paper—the pullback of two occurrence net morphisms can be computed in an efficient and straightforward way.

8 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 interface unfoldings. For the data structures used in 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.

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 component 1 and component 2 is different from the interface between component 1 and component 3 is not straightforward and represents a matter of future investigation.

The task we addressed is closely related to that of [3, 8], so the differences deserve to be pointed out. A first one resides in the notion of system factorisation: [3, 8] 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 [3, 8] to the pullback of interleaving structures (possibly computed through their partial order net 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 [17]. This will only require to verify that the needed properties are satisfied by the category of models at hand.

An approach to distributed unfoldings that uses so-called *augmented processes* is developed in [6, 7]. The main difference resides in the fact that the projection distinguishes more instances of events. In order to clarify this fact, assume that t is a transition of N_0 and thus also of N_1. Suppose t has pre-places in $S_1 - S_0$; then in general, for a given occurrence of t seen by N_0, say e_0, there are several occurrences e_1 of the same transition seen by N_1. These differ for having different histories in N_1, which project to the same history in N_0. In an augmented process such situations are reflected in the projection by keeping different instances of e_0 where the events in N_1 can project. Progressive fusion in that approach leads to a hierarchy of process types, studied in detail in [6, 7], which have greater width w.r.t. conflict than the resulting local unfolding.

Finally, distributed unfolding is orthogonal to the parallelisation of Petri net unfoldings in [10]: that work parallelizes 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. 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.
3. 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.
4. 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.
5. C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Kluwer Academic, 1999.
6. E. Fabre. Factorization of unfoldings for distributed tile systems, part 1: Reduced interaction case. Technical Report 4829, INRIA, May 2003.
7. E. Fabre. Factorization of unfoldings for distributed tile systems, part 2: General case. Technical Report 5186, INRIA, May 2004.
8. 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.
9. 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.
10. 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.
11. S. Mac Lane. *Categories for the working mathematician*. Springer, 1971.
12. 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.
13. M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoret. Comp. Sci.*, 13:85–108, 1981.
14. W. Reisig. *Petri Nets. An Introduction*. Number 4 in EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1982.
15. S. L. Ricker and J. van Schuppen. Decentralized failure diagnosis with asynchronous communication between diagnosers,. In *Proc. of the European Control Conference*, 2001.
16. 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.
17. G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, volume 1. World Scientific, 1997.
18. 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.
19. G. Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.