Approximating the Behaviour of Graph Transformation Systems*

Paolo Baldan[1] and Barbara König[2]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italy
[2] Institut für Informatik, Technische Universität München, Germany

baldan@dsi.unive.it koenigb@in.tum.de

Abstract. We propose a technique for the analysis of graph transformation systems based on the construction of finite structures approximating the behaviour of such systems with arbitrary accuracy. Following a classical approach, one can construct a chain of finite under-approximations (*k-truncations*) of the Winskel's style unfolding of a graph grammar. More interestingly, also a chain of finite over-approximations (*k-coverings*) of the unfolding can be constructed and both chains converge (in a categorical sense) to the full unfolding. The finite over- and under-approximations can be used to check properties of a graph transformation system, like safety and liveness properties, expressed in (meaningful fragments of) the modal μ-calculus. This is done by embedding our approach in the general framework of abstract interpretation.

1 Introduction

Graph transformation systems (GTSs) [28] are a powerful specification formalism for concurrent and distributed systems [10], generalising Petri nets. Along the years their concurrent behaviour has been deeply studied and a consolidated theory of concurrency is now available [27, 11]. Although several semantics of Petri nets, like process and unfolding semantics, have been extended to GTSs (see, e.g., [5, 26, 3, 4]), concerning automated verification, the literature does not contain many contributions to the static analysis of GTSs (see [18, 19]).

Most of the mentioned semantics for GTSs define an operational model of computation, which gives a concrete description of the behaviour of the system in terms of non-effective (e.g., infinite, non-decidable) structures. In this paper, generalising the work in [1], we provide a technique for constructing finite approximations of the behaviour for a class of (hyper)graph transformation systems. We show how one can construct under- and over-approximations of the behaviour of the system. The "accuracy" of such approximations can be fixed and arbitrarily increased in a way that the corresponding chain of (both under- and over-) approximations converges to the exact behaviour.

* Research partially supported by the EC TMR Network GETGRATS, by the IST Project AGILE, and by the MURST project TOSCA.

We concentrate on the unfolding semantics of GTSs, one reason for referring to a concurrent semantics being the fact that it allows to avoid to check all the interleavings of concurrent events. The unfolding construction for GTSs produces a static structure which fully describes the concurrent behaviour of the system, including all possible rewriting steps and their mutual dependencies, as well as all reachable states [26, 4]. However, as already mentioned, the unfolding, being infinite for any non-trivial system, cannot be used directly for verification purposes. Given a *graph grammar*, i.e., a GTS with a start hypergraph, we show how to construct finite structures which can be seen as approximations of the full unfolding of the grammar, at a chosen level k of accuracy.

Under-approximations (k-truncations). The unfolding of a graph grammar $\mathcal{G}$ can be defined categorically as the colimit of its prefixes of finite causal depth. Hence "under-approximations" of the behaviour of $\mathcal{G}$ can be trivially produced by stopping the construction of the unfolding at a finite causal depth k, thus obtaining the so-called *k-truncation* $\mathcal{T}^k(\mathcal{G})$ of the unfolding of $\mathcal{G}$. In the case of Petri nets this is at the basis of the *finite prefix approach*: if the system is finite-state and if the stop condition is suitably chosen, the prefix turns out to be *complete*, i.e., it contains the same information as the full unfolding [22, 12]. In general, for infinite-state systems, any truncation of the unfolding will be just an under-approximation of the behaviour of the system, in the sense that any computation in the truncation can be really performed in the original system, but not vice versa. Nevertheless, finite truncations can still be used to check interesting properties of the grammar, e.g., some liveness properties of the form "eventually A" for a predicate A (see Section 5).

Over-approximations (k-coverings). A more challenging issue is to provide (sensible) over-approximations of the behaviour of a grammar $\mathcal{G}$, i.e., finite approximations of the unfolding which "represent" all computations of the original system (but possibly more). To this aim, generalising [1], we propose an algorithm which, given a graph grammar $\mathcal{G}$, produces a finite structure, called *Petri graph*, consisting of a hypergraph and of a P/T net (possibly not safe or cyclic) over it, which can be seen as an over-approximation of the unfolding. Differently from [1], one can require the approximation to be exact up to a certain causal depth k, thus obtaining the so-called *k-covering* $\mathcal{C}^k(\mathcal{G})$ of the unfolding of $\mathcal{G}$.

The covering $\mathcal{C}^k(\mathcal{G})$ over-approximates the behaviour of $\mathcal{G}$ in the sense that every computation in $\mathcal{G}$ is mapped to a valid computation in $\mathcal{C}^k(\mathcal{G})$. Moreover every hypergraph reachable from the start graph can be mapped homomorphically to (the graphical component of) $\mathcal{C}^k(\mathcal{G})$ and its image is reachable in the Petri graph. Therefore, given a property over graphs reflected by graph morphisms, if it holds for all graphs reachable in the covering $\mathcal{C}^k(\mathcal{G})$ then it also holds for all reachable graphs in $\mathcal{G}$. Important properties of this kind are the non-existence and non-adjacency of edges with specific labels, the absence of certain paths (for checking security properties) or cycles (for checking deadlock-freedom). Temporal properties, such as several safety properties of the form "always A", can be proven directly on the Petri net component of the coverings (see Section 5).

The fact that the unfolding can be approximated with arbitrary high accuracy is formalised by proving that both under- and over-approximations of the unfolding, converge to the full (exact) unfolding. In categorical terms, the unfolding $\mathcal{U}(\mathcal{G})$ of a graph grammar $\mathcal{G}$ can be expressed both as the colimit of the chain of k-truncations $\mathcal{T}^k(\mathcal{G})$ and as the limit of the chain of k-coverings $\mathcal{C}^k(\mathcal{G})$:

$$\begin{tikzcd}
\mathcal{T}^0(\mathcal{G}) \arrow[r] \arrow[rrd] & \mathcal{T}^1(\mathcal{G}) \arrow[r, dotted, no head] \arrow[rd] & \mathcal{T}^k(\mathcal{G}) \arrow[r] \arrow[d] & \mathcal{T}^{k+1}(\mathcal{G}) \arrow[ld] & \cdots \\
 & & \mathcal{U}(\mathcal{G}) \arrow[lld, two heads] \arrow[ld, two heads] \arrow[d, two heads] \arrow[rd, two heads] & & \\
\mathcal{C}^0(\mathcal{G}) & \mathcal{C}^1(\mathcal{G}) \arrow[l, two heads] \arrow[r, dotted, no head] & \mathcal{C}^k(\mathcal{G}) & \mathcal{C}^{k+1}(\mathcal{G}) \arrow[l, two heads] & \cdots
\end{tikzcd}$$

The idea that finite under- and over-approximations can be used for checking properties of a graph grammar $\mathcal{G}$ is enforced by identifying significant fragments of the μ-calculus for which the validity of a formula in some approximation implies the validity of the same formula in the original grammar. Nicely, this is done by viewing our approach as a special case of the general paradigm of abstract interpretation.

2 Hypergraph rewriting, Petri nets and Petri graphs

In this section we first introduce the class of (hyper)graph transformation systems considered in the paper. Then, after recalling some basic notions for Petri nets, we will define Petri graphs, the structure combining hypergraphs and Petri nets, which will be used to represent and approximate the behaviour of GTSs.

2.1 Graph transformation systems

Given a set A we denote by A^* the set of finite strings of elements of A. For $u \in A^*$ we write $|u|$ for the length of u. Moreover, if $f : A \to B$ is a function then $f^* : A^* \to B^*$ denotes its extension to strings. Throughout the paper Λ denotes a fixed set of *labels* and each label $l \in \Lambda$ is associated with an *arity* $ar(l) \in \mathbb{N}$.

Definition 1 (hypergraph). *A (Λ-)*hypergraph *G is a tuple (V_G, E_G, c_G, l_G), where V_G is a set of nodes, E_G is a set of edges, $c_G : E_G \to {V_G}^*$ is a connection function and $l_G : E_G \to \Lambda$ is the labelling function for edges satisfying $ar(l_G(e)) = |c_G(e)|$ for every $e \in E_G$. Nodes are not labelled. A node $v \in V_G$ is called* isolated *if it is not connected to any edge.*

We use rules as in the double-pushout approach [9], with some restrictions.

Definition 2 (rewriting rule). *A graph rewriting rule is a span of injective hypergraph morphisms $r = (L \overset{\varphi_L}{\hookleftarrow} K \overset{\varphi_R}{\hookrightarrow} R)$, where L, K, R are finite hypergraphs. The rule is called* simple *if (i) K is discrete, i.e. it contains no edges, (ii) no two edges in the left-hand side L have the same label, (iii) the morphism φ_L is bijective on nodes, (iv) V_L does not contain isolated nodes.*

Hereafter we will restrict to simple rules. A simple rule can delete and produce but not preserve edges, while nodes cannot be deleted (conditions (i) and (iii)). Moreover, it cannot consume two edges with the same label and its left-hand side must be connected (conditions (ii) and (iv)). These restrictions are mainly aimed at simplifying the presentation. Only (iii), which allows to apply a rule without checking the dangling condition, could require serious technical complications to be removed (but observe that deletion of nodes can be simulated considering graphs up to isolated nodes and leaving a node isolated instead of deleting it).

To simplify the notation, in the following we will assume that for any rule $r = (L \overset{\varphi_L}{\hookleftarrow} K \overset{\varphi_R}{\hookrightarrow} R)$, the morphisms φ_L and φ_R are (set-theoretical) inclusions and that $K = L \cap R$ (componentwise). Furthermore the components of a rule r will be denoted by L_r, K_r and R_r.

Definition 3 (hypergraph rewriting). *Let r be a rewriting rule. A* match *of r in a hypergraph G is any morphism $\varphi : L_r \to G$. In this case we write $G \Rightarrow_{r,\varphi} H$ or simply $G \Rightarrow_r H$, if there exists a double-pushout diagram*

$$\begin{tikzcd}
L_r \arrow[d, "\varphi"'] & K_r \arrow[l, hook'] \arrow[r, hook] \arrow[d] & R_r \arrow[d] \\
G & D \arrow[l, hook'] \arrow[r, hook] & H
\end{tikzcd}$$

Given a graph transformation system (GTS)*, i.e., a finite set of rules $\mathcal{R}$, we write $G \Rightarrow_{\mathcal{R}} H$ if $G \Rightarrow_r H$ for some $r \in \mathcal{R}$. Moreover $\Rightarrow^*_{\mathcal{R}}$ denotes the transitive closure of $\Rightarrow_{\mathcal{R}}$. A* GTS *with a (finite) start graph $\mathcal{G} = (\mathcal{R}, G_{\mathcal{R}})$ is called a* graph grammar*.*

2.2 Petri nets

We fix some basic notation for Petri nets [25, 23]. Given a set A we will denote by $A^{\oplus}$ the free commutative monoid over A (*multisets* over A). Given a function $f : A \to B$, by $f^{\oplus} : A^{\oplus} \to B^{\oplus}$ we denote its monoidal extension.

Definition 4 (Petri net). *Let A be a finite set of action labels. An A-*labelled Petri net *is a tuple $N = (S, T, {}^\bullet(), ()^\bullet, p)$ where S is a set of places, T is a set of transitions, ${}^\bullet(), ()^\bullet : T \to S^{\oplus}$ assign to each transition its pre-set and post-set and $p : T \to A$ assigns an action label to each transition.*

The Petri net is called irredundant *if there are no distinct transitions with the same label and pre-set, i.e., if for any $t, t' \in T$*

$$p(t) = p(t') \ \wedge \ {}^\bullet t = {}^\bullet t' \quad \Rightarrow \quad t = t'. \tag{1}$$

A marked Petri net *is a pair (N, m_N), where N is a Petri net and $m_N \in S^{\oplus}$ is the initial marking.*

The irredundancy condition (1) aims at avoiding the presence of multiple events, indistinguishable for what regards the behaviour of the system. Hereafter *all* the considered Petri nets will be assumed irredundant, unless stated otherwise.

Definition 5 (causality relation). *Let N be a (marked) Petri net. The* causality relation *$<_N$ over N is the least transitive relation such that, for any $t \in T$, $s \in S$, we have (i) $s <_N t$ if $s \in {}^\bullet t$ and (ii) $t <_N s$ if $s \in t^\bullet$.*

2.3 Petri graphs

Petri graphs, as introduced in [1], are structures consisting of a hypergraph and of a Petri net whose places are the edges of the graph.

Definition 6 (Petri graph). *Let $\mathcal{R}$ be a* GTS. *A Petri graph (over $\mathcal{R}$) is a tuple $P = (G, N, \mu)$ where G is a hypergraph, $N = (E_G, T_N, {}^\bullet(), ()^\bullet, p_N)$ is an $\mathcal{R}$-labelled Petri net with edges of G as places, and μ associates to each transition $t \in T_N$, with $p_N(t) = r$, a hypergraph morphism $\mu(t) : L_r \cup R_r \to G$ such that*

$$^\bullet t = \mu(t)^\oplus(E_{L_r}) \ \wedge \ t^\bullet = \mu(t)^\oplus(E_{R_r}) \tag{2}$$

A Petri graph for a grammar $(\mathcal{R}, G_\mathcal{R})$ is a pair (P, ι) where $P = (G, N, \mu)$ is a Petri graph for $\mathcal{R}$ and $\iota : G_\mathcal{R} \to G$ is a graph morphism. The multiset $\iota^\oplus(E_{G_\mathcal{R}})$ is called initial marking *of the Petri graph. A marking $m \in E_G{}^\oplus$ is called* reachable (coverable) *in (P, ι) if it is reachable (coverable) in the underlying Petri net.*

Condition (2) allow to interpret transitions in the net as "occurrences" of rules in $\mathcal{R}$. More precisely, if $p_N(t) = r$ and $\mu(t) : L_r \cup R_r \to G$ is the morphism associated to the transition, then $\mu(t)_{|L} : L_r \to G$ must be a match of r in G such that the image of the edges of L_r in G coincides with the pre-set of t. Then, the graph items resulting from the application of r must be already in G, and the corresponding edges must coincide with the post-set of t. This is formalised by the condition over the image through $\mu(t)$ of the edges of R_r. For an example see Section 5, where Fig. 2 presents two Petri graphs for the GTS in Fig. 1.

A safe marking m of a Petri graph $P = (G, N, \mu)$ is intended to represent the subgraph of G consisting of the edges in m and of the nodes attached to these edges. For a general non-safe marking edges with k tokens will result in k "parallel" edges. This is formalised in the next definition.

Definition 7. *Let $P = (G, N, \mu)$ be a Petri graph. Given a hypergraph morphism $\varphi : G' \to G$ injective on nodes, we say that the marking $\varphi^\oplus(E_{G'})$* generates *the graph G'.*

In the following we will often confuse a marking of a Petri graph with its generated graph, and say, e.g., that a given graph is reachable in a Petri graph.

Every hypergraph G can be considered as a Petri graph $[G] = (G, N, \mu)$ for $\mathcal{R}$, by taking N as the net with $S_N = E_G$ and no transitions. Similarly, $G_\mathcal{R}$ can be seen as a Petri graph for $(\mathcal{R}, G_\mathcal{R})$ by taking as $\iota : G_\mathcal{R} \to G_\mathcal{R}$ the identity.

Definition 8 (category of Petri graphs). *A* Petri graph morphism *is a pair $\psi = (\varphi, \tau) : (G, N, \mu) \to (G', N', \mu')$ where*

- *$\varphi : G \to G'$ is a hypergraph morphism;*
- *$\tau : T_N \to T_{N'}$ is a mapping such that for every $t \in T_N$, ${}^\bullet\tau(t) = \varphi^\oplus({}^\bullet t)$ and $\tau(t)^\bullet = \varphi^\oplus(t^\bullet)$, and $p_{N'} \circ \tau = p_N$.*
- *for every $t \in T_N$, $\mu'(\tau(t)) = \varphi \circ \mu(t)$.*

The category of Petri graphs and Petri graph morphisms is denoted by PG.

It is possible to show that the category PG is finitely cocomplete, i.e. it contains all finite colimits. In particular we will later make use of pushouts and coequalizers to define unfolding and folding operations.

3 Unfolding and under-approximations

In this section we define the unfolding of a graph grammar. Following a common approach in the literature (see, e.g., [26, 29]) the unfolding is defined as the limit (actually, the categorical colimit) of the chain of its finite prefixes, each of which can be seen as an *under-approximation* of the behaviour of the system.

The finite prefixes of the unfolding are constructed inductively beginning from the start graph of the grammar and performing, at each stage, all the possible basic unfolding steps, until the given causal depth is reached. A basic step roughly consists of the "partial" application of a rule to a match, which does not delete the left-hand side, but only records the new graph item generated in the rewriting process and the rule occurrence.

To formally define a basic step we need to fix some notation. Given a transition t and a rule r we will denote by $P(t,r)$ the Petri graph $(L_r \cup R_r, N, \mu)$ where $N = (E_{L_r \cup R_r}, \{t\}, {}^\bullet t = E_{L_r}, t^\bullet = E_{R_r}, p_N(t) = r)$ and $\mu(t) = id_{L_r \cup R_r}$. By $\emptyset$ we denote a function with an empty set as domain.

Definition 9 (unfolding operation). *Let $P = (G, N, \mu)$ be a Petri graph for a* GTS *$\mathcal{R}$. Let $r \in \mathcal{R}$ be a rule and let $\varphi : L_r \to G$ be a match of r in G. The* unfolding *of P with rule r at match φ, denoted by* $\mathsf{unf}(P, r, \varphi)$*, is the Petri graph obtained as pushout of $(\varphi, \emptyset) : [L_r] \to P$ and $(id_{L_r}, \emptyset) : [L_r] \to P(t,r)$.*

If (P, ι) is a Petri graph for a graph grammar $(\mathcal{R}, G_{\mathcal{R}})$, in the same situation, we define $\mathsf{unf}((P,\iota), r, \varphi) = (P', \psi \circ \iota)$ *where $(\psi, \tau) : P \to P'$ is the* PG *morphism generated by the pushout.*

We need to define the depth of an item in a Petri graph. We start with a definition of depth over Petri nets. To deal with the presence of causal cycles it is convenient to define several depth functions, each one measuring the depth of an item up to a fixed level k. Consider the monoid $\mathbb{M}_k = (\{0, \ldots, k\}, +)$, where for $m, n \in \{0, \ldots, k\}$, $m + n$ is ordinary addition if $m + n \leq k$ and $m + n = k$ otherwise.

Definition 10 (depth in a Petri net). *Let N be a Petri net. We define a function $D : (S_N \cup T_N \to \mathbb{M}_k) \to (S_N \cup T_N \to \mathbb{M}_k)$ as follows:*

$$D(d)(x) = \max\{d(s) \mid s \in S_N \ \wedge \ s < x\} + 1.$$

Then the function $depth_k : S_N \cup T_N \to \mathbb{M}_k$, assigning depth information to every Petri net item is the least fixed point of D.

The function $depth_k$ assigns to each item x of a Petri net its causal depth, i.e., the length h of the maximal chain of causally related items leading from the initial marking to x, when $h \leq k$ and k otherwise. Note that an item x located in a causality cycle has always maximal depth, i.e., $depth_k(x) = k$ for any k.

The definition generalises to Petri graphs in a straightforward way: places become edges and the depth of a node v is defined as the maximal depth of rules r where v appears in $R_r \setminus L_r$ (intuitively, of rules which can "generate" node v).

Definition 11 (depth of items in a Petri graph). *Let (P, ι) be a Petri graph with $P = (G, N, \mu)$. For any k the function $depth_k : E_G \cup T_N \to \mathbb{M}_k$ is defined as in Definition 10. This function is extended to nodes by defining, for $v \in V_G$*

$$depth_k(v) = \max\{depth_k(t) \mid p_N(t) = r \ \wedge \ v \in \mu(t)(V_{R_r} \backslash V_{L_r})\}$$

The prefixes of the unfolding of a graph grammar up to a given causal depth k are defined by the following algorithm.

Definition 12 (k-truncation). *Let $k \in \mathbb{N}$ and let $\mathcal{G} = (\mathcal{R}, G_{\mathcal{R}})$ be a graph grammar. The algorithm generates a sequence $(P_i, \iota_i)_{i \in \mathbb{N}}$ of Petri graphs.*

(Step 0) *Initialise $(P_0, \iota_0) = ([G_{\mathcal{R}}], id_{G_{\mathcal{R}}})$.*

(Step $i+1$) *Let (P_i, ι_i), with $P_i = (G_i, N_i, \mu_i)$, be the Petri graph produced at step i.*

$\star$ *Unfolding: Find a rule r in $\mathcal{R}$ and a match $\varphi : L_r \to G_i$ such that*
- *$\varphi^{\oplus}(E_{L_r})$ is a coverable marking in P_i;*
- *there is no transition $t \in T_{N_i}$ such that ${}^{\bullet}t = \varphi^{\oplus}(E_{L_r})$ and $p_{N_i}(t) = r$;*
- *for all $x \in \varphi(L_r)$ it holds that $depth_k(x) \neq k$.*

Then set $(P_{i+1}, \iota_{i+1}) = \mathsf{unf}((P_i, \iota_i), r, \varphi)$.

*If no unfolding step can be performed, the algorithm stops. The resulting Petri graph (P_i, ι_i) is called k-*truncation *of the unfolding of $\mathcal{G}$ and denoted by $\mathcal{T}^k(\mathcal{G})$.*

It can be easily proven that the unfolding procedure described above is terminating and confluent, and thus that $\mathcal{T}^k(\mathcal{G})$ is well-defined. Furthermore, $\mathcal{T}^{k+1}(\mathcal{G})$ can be obtained from $\mathcal{T}^k(\mathcal{G})$ by performing only the unfolding steps which involve items of depth k. This gives a uniquely determined embedding $\lambda_k : \mathcal{T}^k(\mathcal{G}) \to \mathcal{T}^{k+1}(\mathcal{G})$ for any $k \in \mathbb{N}$. The diagram $\mathcal{T}^0(\mathcal{G}) \xrightarrow{\lambda_0} \ldots \mathcal{T}^k(\mathcal{G}) \xrightarrow{\lambda_k} \mathcal{T}^{k+1}(\mathcal{G}) \xrightarrow{\lambda_{k+1}} \ldots$ is called the *truncation tower.*

The next definition introduces the full unfolding of a graph grammar as colimit of its finite truncations (which can be shown to exist).

Definition 13 (unfolding as colimit of the k-truncations). *The* (full) unfolding *$\mathcal{U}(\mathcal{G})$ of a graph grammar $\mathcal{G}$ is the colimit of the truncation tower.*

The proposition below clarifies in which sense the unfolding represents the behaviour of the original grammar: any graph reachable in a graph grammar can be mapped *isomorphically* to a reachable subgraph of its unfolding, and, vice versa, any reachable subgraph of the unfolding is the isomorphic image of a reachable graph in the original grammar. Furthermore steps in the original grammar correspond to steps in the unfolding [26, 4].

Proposition 14. *Let $\mathcal{G} = (\mathcal{R}, G_{\mathcal{R}})$ be a graph grammar and let $\mathcal{U}(\mathcal{G}) = (U, N, \mu)$ be its unfolding. Then for every graph G we have $G_{\mathcal{R}} \Rightarrow^*_{\mathcal{R}} G$ iff there exists an injective morphism $\varphi_G : G \to U$ and the marking ${\varphi_G}^{\oplus}(E_G)$ is reachable in $\mathcal{U}(\mathcal{G})$. Furthermore, in the situation above if $G \Rightarrow_{\mathcal{R}} G'$ then ${\varphi_G}^{\oplus}(E_G) \xrightarrow{t} {\varphi_{G'}}^{\oplus}(E_{G'})$ for a suitable transition t in $\mathcal{U}(\mathcal{G})$. And if ${\varphi_G}^{\oplus}(E_G) \xrightarrow{t} m$ for some marking m, then there exists a graph G' such that $G \Rightarrow_{\mathcal{R}} G'$ and $m = {\varphi_{G'}}^{\oplus}(E_{G'})$.*

Clearly, k-truncations provide, in general, only under-approximations of the behaviour of the original grammar $\mathcal{G}$, i.e., only one implication of Proposition 14 holds: any graph reachable in $\mathcal{T}^k(\mathcal{G})$ is mapped isomorphically to a graph reachable in $\mathcal{G}$ and any valid computation in $\mathcal{T}^k(\mathcal{G})$ corresponds to a valid derivation sequence in $\mathcal{G}$, but, in general, not vice versa. Still, as we will see in Section 5, k-truncations can be useful for proving properties of the original grammar.

4 Folding and over-approximations

In this section we define an algorithm which, given a graph grammar $\mathcal{G}$ and a level of accuracy k, produces a finite Petri graph $\mathcal{C}^k(\mathcal{G})$, called *$k$-covering*, which can be seen as an over-approximation of the behaviour of the grammar $\mathcal{G}$.

We have already mentioned that the full unfolding is usually infinite, also for finite-state systems. To obtain a finite over-approximation we modify the unfolding procedure by considering, besides the *unfolding rule*, also a *folding rule* which allows us to "merge" two occurrences of the left-hand side of a rule whenever they are, in a sense made precise later, one causally dependent on the other. Intuitively, the presence of such two occurrences of a left-hand side reveals a cyclic behaviour and applying the folding rule one avoids to unfold the corresponding infinite path. While guaranteeing finiteness, the folding operation causes a loss of information in a way that the resulting structure over-approximates the behaviour of the original system: every graph reachable in the original grammar $\mathcal{G}$ corresponds to a marking which is reachable in the covering and every valid derivation in $\mathcal{G}$ corresponds to a valid firing sequence in the covering (but not vice versa).

In order to compute better over-approximations of the behaviour the idea is to delay folding steps, constraining the algorithm to apply only unfolding steps until a given causal depth is reached. Roughly, this is obtained by "freezing" an initial part of the approximated unfolding, up to a given causal depth k, and by allowing only unfolding and no folding steps to affect that part. The resulting over-approximation $\mathcal{C}^k(\mathcal{G})$ is "exact" up to causal depth k, in the sense that any graph reachable in $\mathcal{G}$ in less than k steps will have a reachable *isomorphic* image in $\mathcal{C}^k(\mathcal{G})$. Instead, graphs which are reachable in a larger number of steps, in general, will be mapped homomorphically in $\mathcal{C}^k(\mathcal{G})$ (still to a reachable graph).

In this way one can obtain arbitrarily accurate approximations, a fact which is enforced by proving that the chain of k-coverings of a grammar $\mathcal{G}$ converges to the full (possibly infinite) unfolding $\mathcal{U}(\mathcal{G})$. In categorical terms, $\mathcal{U}(\mathcal{G})$ turns out to be the limit of the chain of coverings in a suitable subcategory of Petri graphs.

4.1 Computing k-coverings

A basic definition needed to introduce k-coverings is that of a folding operation. Intuitively, it allows to merge two matches of the same rule in a Petri graph.

Definition 15 (folding operation). *Let $P = (G, N, \mu)$ be a Petri graph for a* GTS *$\mathcal{R}$. Let $r \in \mathcal{R}$ be a rule and let $\varphi', \varphi : L_r \to G$ be matches of r in G. The* folding *of P at the matches φ', φ, denoted $\mathsf{fold}(P, r, \varphi', \varphi) = P'$, is the Petri graph P' obtained as the coequalizer of $(\varphi, \emptyset), (\varphi', \emptyset) : [L_r] \to P$ in category* PG.

If (P, ι) is a Petri graph for a graph grammar $(\mathcal{R}, G_{\mathcal{R}})$, in the same situation, we define $\mathsf{fold}((P, \iota), r, \varphi', \varphi) = (P', \psi \circ \iota)$ where $(\psi, \tau) : P \to P'$ is the PG *morphism generated by the coequalizer.*

The algorithm which produces the k-covering $\mathcal{C}^k(\mathcal{G})$ generates a sequence of Petri graphs, beginning from the start graph of $\mathcal{G}$ and applying, non-deterministically, at each step, a folding or unfolding operation, until none of such steps is admitted. Folding steps will be applied only at depth k or greater. Note that as soon as folding steps are applied, the Petri graph will contain cycles.

Definition 16 (k-covering). *Let $\mathcal{G} = (\mathcal{R}, G_{\mathcal{R}})$ be a graph grammar and let $k \in \mathbb{N}$. The algorithm generates a sequence $(P_i, \iota_i)_{i \in \mathbb{N}}$ of Petri graphs, as follows.*

(Step 0) *Initialise $(P_0, \iota_0) = ([G_{\mathcal{R}}], id_{G_{\mathcal{R}}})$.*

(Step $i+1$) *Let (P_i, ι_i), with $P_i = (G_i, N_i, \mu_i)$, be the Petri graph produced at step i. Choose non-deterministically one of the following actions*

$\star$ *Folding: Find a rule r in $\mathcal{R}$ and two different matches $\varphi', \varphi : L_r \to G_i$ of r such that*

- *$\varphi^{\oplus}(E_{L_r})$ is a coverable marking in P_i;*
- *there exists a transition $t \in T_{N_i}$ such that*

$$p_{N_i}(t) = r \ \wedge\ {}^{\bullet}t = {\varphi'}^{\oplus}(E_{L_r}) \ \wedge\ \forall e \in \varphi^{\oplus}(E_{L_r}) : (e \in {}^{\bullet}t \ \vee\ t <_{N_i} e) \quad (3)$$

- *for every edge or node $x \in E_{L_r} \cup V_{L_r}$ it holds that*

$$\varphi(x) = \varphi'(x) \ \vee\ depth_k(\varphi(x)) = depth_k(\varphi'(x)) = k. \quad (4)$$

Then set $(P_{i+1}, \iota_{i+1}) = \mathsf{fold}((P_i, \iota_i), r, \varphi', \varphi)$.

$\star$ *Unfolding: Find a rule r in $\mathcal{R}$ and a match $\varphi : L_r \to G_i$ such that*

- *$\varphi^{\oplus}(E_{L_r})$ is a coverable marking in P_i;*
- *there is no transition $t \in T_{N_i}$ such that ${}^{\bullet}t = \varphi^{\oplus}(E_{L_r})$ and $p_{N_i}(t) = r$;*
- *there is no other match $\varphi' : L_r \to G_i$ satisfying the folding condition.*

Then set $(P_{i+1}, \iota_{i+1}) = \mathsf{unf}((P_i, \iota_i), r, \varphi)$.

*If no folding or unfolding step can be performed, the algorithm terminates. The resulting Petri graph (P_i, ι_i) is called k-*covering *of the unfolding of $\mathcal{G}$ and denoted by $\mathcal{C}^k(\mathcal{G})$.*

Condition (3) basically states that we can fold two matches of a rule r whenever the first one has been already unfolded producing a transition t, and the second match depends on the first one, in the sense that any edge in the second match is already in the first one or causally depends on t. Roughly, the idea is that we should not unfold a left-hand side again, if we have already done the same unfolding step in its past, since this might lead to infinitely many steps. There are some similarities, to be further investigated, with the work in [14] where the sets of descendants and of normal forms of term rewriting systems are approximated by constructing an approximation automaton. Additionally, by Condition (4) only items of depth k can be merged, in a way that the prefix up to depth k of the unfolding is not involved in any folding operations. Actually some items of depth less than k can be part of a folding operation, but they must be left unchanged by the step.

4.2 Correctness, termination and confluence

We first show that the computed Petri graph $\mathcal{C}^k(\mathcal{G})$ gives an over-approximation of the behaviour of the given graph grammar, exact up to causal depth k. More precisely we prove that for any graph reachable in $\mathcal{G}$, there is a morphism into the covering $\mathcal{C}^k(\mathcal{G})$ such that the image of its edge set corresponds to a reachable marking. Furthermore, if a graph is reachable in $\mathcal{G}$ in less than k steps, then it will be mapped isomorphically to to (the graphical component) of $\mathcal{C}^k(\mathcal{G})$.

Proposition 17 (correctness). *Let $\mathcal{G} = (\mathcal{R}, G_\mathcal{R})$ be a graph grammar and assume that the algorithm computing the k-covering terminates producing the Petri graph $\mathcal{C}^k(\mathcal{G}) = ((U, N, \mu), \iota)$. Then for every graph G*

i) if $G_\mathcal{R} \Rightarrow^*_\mathcal{R} G$ there exists a morphism $\varphi_G : G \to U$ and the marking ${\varphi_G}^\oplus(E_G)$ is reachable in $\mathcal{C}^k(\mathcal{G})$. Furthermore, if $G \Rightarrow_\mathcal{R} G'$ then ${\varphi_G}^\oplus(E_G) \xrightarrow{t} {\varphi_{G'}}^\oplus(E_{G'})$ for a suitable transition t in $\mathcal{C}^k(\mathcal{G})$.

ii) If $G_\mathcal{R} \Rightarrow^*_\mathcal{R} G$ with a (possibly parallel) derivation of length less than k then there exists an injective morphism $\varphi_G : G \to U$ such that the marking ${\varphi_G}^\oplus(E_G)$ is reachable in $\mathcal{C}^k(\mathcal{G})$ and $\max\{depth_k(x)) \mid x \in G\} < k$, and vice versa. Furthermore if ${\varphi_G}^\oplus(E_G) \xrightarrow{t} m$ for some transition t, then there exists a graph G' such that $G \Rightarrow_\mathcal{R} G'$ and $m = {\varphi_{G'}}^\oplus(E_{G'})$.

It is not obvious at first glance that the algorithm computing the k-covering always terminates. To prove termination we rely on the corresponding result in [1] where we show that it is not possible to perform infinitely many unfolding steps, without having the folding condition satisfied at some stage.

Proposition 18 (termination). *The algorithm computing the k-covering (see Definition 16) terminates for every graph grammar $\mathcal{G}$ and every $k \in \mathbb{N}$.*

In order to prove that the algorithm produces a uniquely determined result, independently of the order in which folding and unfolding steps are applied, we can show that the rewriting relation on Petri graphs induced by folding and unfolding steps is locally confluent. By the Diamond Lemma [8], for a rewriting system local confluence and termination imply confluence.

Proposition 19 (confluence). *For any input grammar $\mathcal{G}$ and $k \in \mathbb{N}$ the algorithm computing the k-covering terminates with a result $\mathcal{C}^k(\mathcal{G})$ unique up to isomorphism.*

4.3 Full unfolding as limit of the coverings

The fact that folding and unfolding operations are given in terms of colimits allows us to define, for any k, a (uniquely determined) Petri graph morphism $\upsilon_k : \mathcal{C}^{k+1}(\mathcal{G}) \to \mathcal{C}^k(\mathcal{G})$. The diagram $\mathcal{C}^0(\mathcal{G}) \overset{\upsilon_0}{\leftarrow} \ldots \mathcal{C}^k(\mathcal{G}) \overset{\upsilon_k}{\leftarrow} \mathcal{C}^{k+1}(\mathcal{G}) \overset{\upsilon_{k+1}}{\leftarrow} \ldots$ is called the *covering tower.*

The next proposition presents a central result of this paper. For technical reasons we consider the full subcategory PG^* of PG having as objects Petri graphs in which every edge is coverable and every transition can be fired.

Proposition 20 (unfolding as limit of the coverings). *The limit in the category PG^* of the covering tower $\mathcal{C}^0(\mathcal{G}) \overset{\upsilon_0}{\leftarrow} \ldots \mathcal{C}^k(\mathcal{G}) \overset{\upsilon_k}{\leftarrow} \mathcal{C}^{k+1}(\mathcal{G}) \overset{\upsilon_{k+1}}{\leftarrow} \ldots$ is the full unfolding $\mathcal{U}(\mathcal{G})$ of the graph grammar.*

5 Checking Temporal Properties

In this section we illustrate how our technique can be seen as a specific instance of abstract interpretation [7, 17]. Embedding our work into this context we can resort to some results from [20], thus identifying classes of temporal properties (μ-calculus formulae) which, being preserved/reflected by abstractions, can be studied over suitable approximations of a GTS.

We recall some concepts from [20], the more basic one being the formalisation of abstraction given in terms of Galois connections (over powerset lattices).

Definition 21 (Galois connection). *Let Q_1 and Q_2 be two sets of states. A* Galois connection *from $\mathcal{P}(Q_1)$ to $\mathcal{P}(Q_2)$ is a pair of monotonic functions (α, γ), with $\alpha : \mathcal{P}(Q_1) \to \mathcal{P}(Q_2)$* (abstraction) *and $\gamma : \mathcal{P}(Q_2) \to \mathcal{P}(Q_1)$* (concretization)*, such that $id_{Q_1} \subseteq \gamma \circ \alpha$ and $\alpha \circ \gamma \subseteq id_{Q_2}$.*

Next we introduce $\langle \alpha, \gamma \rangle$-simulations which turn out to coincide with simulations in the sense of Milner (see [20] for details).

Definition 22 ($\langle \alpha, \gamma \rangle$-simulation). *Let $T_i = (Q_i, \to_i)$ with $i \in \{1, 2\}$ be transition systems, where Q_i is a set of* states *and $\to_i \subseteq Q_i \times Q_i$ is the* transition relation*. Let furthermore (α, γ) be a Galois connection from $\mathcal{P}(Q_1)$ to $\mathcal{P}(Q_2)$.*

*We say that T_2 $\langle \alpha, \gamma \rangle$-*simulates *$T_2$, written $T_1 \sqsubseteq_{\langle \alpha, \gamma \rangle} T_2$, if $\alpha \circ pre[\to_1] \circ \gamma \subseteq pre[\to_2]$, where the function $pre[\to_i] : \mathcal{P}(Q_i) \to \mathcal{P}(Q_i)$ is defined by $pre[\to_i](Q) = \{q \in Q_i \mid \exists\, q' \in Q : q \to_i q'\}$.*

Let T_1, T_2 be transition systems and let $\varphi : T_1 \to T_2$ be a *transition system morphism*, i.e., a function $\varphi : Q_1 \to Q_2$ such that such that for any $q, q' \in Q_1$ if $q \to_1 q'$ then $\varphi(q) \to_2 \varphi(q')$ (in other words, φ is a special kind of simulation).

Then, it can be easily seen that the pair (φ, φ^{-1}) is a Galois connection and furthermore $T_1 \sqsubseteq_{\langle \varphi, \varphi^{-1} \rangle} T_2$.

We next discuss how our under- and over-approximations of the behaviour of a graph grammar can be interpreted in this context. First observe that, a Petri graph (P, ι), with $P = (G, N, \mu)$, can be associated with a transition system $\mathbb{M}_{(P,\iota)}$, having reachable markings (multi-sets of edges) as states and the firing relation of the underlying Petri net N as transition relation. Alternatively we can consider the transition system, $\mathbb{G}_{(P,\iota)}$, where states are graphs (generated by the reachable markings, in the sense of Definition 7) and the transition relation is again induced by the firing relation of N.

Let $\mathcal{G}$ be a graph grammar and consider the full unfolding $\mathcal{U}(\mathcal{G})$, the k-truncations $\mathcal{T}^k(\mathcal{G})$ and the k-coverings $\mathcal{C}^k(\mathcal{G})$. Since by Definition 13 and Proposition 20 the full unfolding is the colimit of the truncations and the limit of the coverings, we have (unique) morphisms $\eta_k : \mathcal{T}^k(\mathcal{G}) \to \mathcal{U}(\mathcal{G})$ and $\theta_k : \mathcal{U}(\mathcal{G}) \to \mathcal{C}^k(\mathcal{G})$, which can be regarded as functions from sets of markings to sets of markings and furthermore they are morphisms between the transition systems of the underlying Petri nets. Hence we have the following result.

Proposition 23. *Let $\mathcal{G}$ be a graph grammar. Then (η_k, η_k^{-1}) and $(\theta_k, \theta_k^{-1})$ are Galois connections and $\mathbb{M}_{\mathcal{T}^k(\mathcal{G})} \sqsubseteq_{\langle \eta_k, \eta_k^{-1} \rangle} \mathbb{M}_{\mathcal{U}(\mathcal{G})} \sqsubseteq_{\langle \theta_k, \theta_k^{-1} \rangle} \mathbb{M}_{\mathcal{C}^k(\mathcal{G})}$.*

Modal μ-calculus. One of the central results of [20] is the preservation and reflection of modal μ-calculus formulae on transitions systems. Recall that the modal μ-calculus is a temporal logic enriched with fixed-point operators. The syntax of μ-calculus formulae is the following:

$$f ::= \; A \mid X \mid \Diamond f \mid \Box f \mid \neg f \mid f_1 \vee f_2 \mid f_1 \wedge f_2 \mid \mu X.f \mid \nu X.f$$

where $A \in \mathcal{A}$ are *atomic propositions* and $X \in \mathcal{X}$ are *proposition variables.* The formulae are evaluated over a transition system $T = (Q, \to)$, with respect to an *interpretation* $I : \mathcal{A} \to \mathcal{P}(Q)$, associating to any atomic proposition $A \in \mathcal{A}$ the set of states $I(A)$ where it holds. A formula $\Diamond f$ / $\Box f$ holds in a state q if some / any single step leads to a state where f holds. The connectives $\neg$,$\vee$, $\wedge$ are interpreted in the usual way. The formulae $\mu X.f$ and $\nu X.f$ represent the *least* and *greatest fixed point*, respectively. We write $q \models^I f$ to mean that the (closed) formula f holds in the state q, under the interpretation I. We say that a transition system T *satisfies a (closed) formula f under an interpretation I*, written $T \models^I f$, if $q_0 \models^I f$ where q_0 is the initial state of T.

The fragment of the modal μ-calculus without negation and box operator is denoted by $\Diamond L_\mu$. By dropping negation and the diamond operator we obtain the fragment $\Box L_\mu$. Some typical *liveness* properties of the form "eventually A" (i.e., $\mu X.(A \vee \Diamond X)$) can be expressed in the fragment $\Diamond L_\mu$, whereas some typical *safety* properties of the form "always A" (i.e., $\nu X.(A \wedge \Box X)$) can be expressed in the fragment $\Box L_\mu$. However, while for linear time there exists a syntactic characterization of liveness and safety properties [24], in the case of branching time there is not yet any established definition of liveness and safety [21].

Let us come back to graph transformation systems, where atomic propositions stand for graph properties, i.e., for sets of graphs. Let (P, ι) be a Petri graph and let f be a μ-calculus formula over a set of atomic propositions $\mathcal{A}$. Assume that I_m and I_g are interpretations of $\mathcal{A}$ over $\mathbb{M}_{(P,\iota)}$ and $\mathbb{G}_{(P,\iota)}$, respectively, such that, for any $A \in \mathcal{A}$, any marking m and graph $G(m)$ generated by m

$$m \in I_m(A) \quad \mathit{iff} \quad G(m) \in I_g(A). \tag{5}$$

Then it is immediate to see that $\mathbb{M}_{(P,\iota)} \models^{I_m} f$ if and only if $\mathbb{G}_{(P,\iota)} \models^{I_g} f$. Furthermore, given a graph grammar $\mathcal{G}$, seen as a transition system in the obvious way, by Proposition 14 it follows that $\mathcal{G} \models^{I_g} f$ if and only if $\mathbb{G}_{\mathcal{U}(\mathcal{G})} \models^{I_m} f$.

Using the above observations and exploiting the preservation and reflection properties in [20] we can obtain the following result. We say that a set Gr of hypergraphs is *preserved* by graph morphisms whenever the existence of a morphism $\varphi : G \to G'$ with $G \in Gr$ implies $G' \in Gr$. Symmetrically, Gr is *reflected* by graph morphisms whenever the existence of a morphism $\varphi : G \to G'$ with $G' \in Gr$ implies $G \in Gr$.

Corollary 24. *Let $\mathcal{G} = (\mathcal{R}, G_{\mathcal{R}})$, let f be a μ-calculus formula over a set of atomic propositions $\mathcal{A}$. Let I_m and I_g be interpretations satisfying (5). Then*

– *If $f \in \Diamond L_\mu$, $\mathbb{M}_{\mathcal{T}^k(\mathcal{G})} \models^{I_m} f$ and every set $I_g(A)$ is preserved by hypergraph morphisms, then $\mathcal{G} \models^{I_g} f$.*
– *If $f \in \Box L_\mu$, $\mathbb{M}_{\mathcal{C}^k(\mathcal{G})} \models^{I_m} f$ and every set $I_g(A)$ is reflected by hypergraph morphisms, then $\mathcal{G} \models^{I_g} f$.*

We have shown how to reduce the analysis of the full transition system of a graph grammar to the analysis of simpler transition systems, generated by Petri nets (underlying Petri graphs). These transition systems might still have infinitely many states, but there are several decidability results for the modal μ-calculus and other forms of temporal logics [13, 15, 16].

Example. Let us consider the simple graph grammar $\mathcal{S}$ in Fig. 1, where edge labels have the following meaning: C (connections), S_{pub} (public servers), S_{prv} (private servers), P_{int} (internal processes) and P_{ext} (external processes). Internal processes can wander around the network and public servers can extend the network by creating new connections. Our aim is to show that the external process is never connected to a private server and thus has access to classified data. That is, we want to show that the following logical formula is satisfied by the graph transformation system: $f = \nu X.(A \;\wedge\; \Box X)$ where the atomic proposition A holds for all the graphs in $Gr_A = \{G \mid \forall e_1, e_2 \in E_G.(l_G(e_1) = S_{prv} \;\wedge\; l_G(e_2) = P_{ext} \;\Rightarrow\; c_G(e_1) \neq c_G(e_2))\}$ (it always holds that whenever a private server and an external process appear in a graph, then they are not connected to the same node). One can easily show that Gr_A is reflected by hypergraph morphisms.

Applying the algorithm in Definition 16 to the graph grammar $\mathcal{S}$ to compute the 0-covering $\mathcal{C}^0(\mathcal{S})$, we obtain the left-hand Petri graph in Fig. 2. Observe

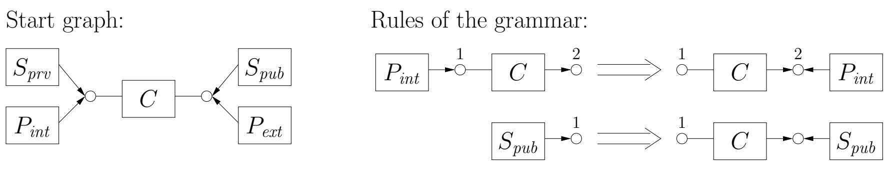

Fig. 1. The example graph grammar $\mathcal{S}$.

that the formula f is not satisfied by this covering, since A is invalid already for the initial marking. Hence this gives us no indication whether or not the formula holds for $\mathcal{S}$. Therefore we try and compute the 1-covering and get the Petri graph on the right-hand side of Fig. 2. Now we can establish that f holds just by looking at the graph structure of the 1-covering $\mathcal{C}^1(\mathcal{S})$: edges of the form S_{prv} and of the form P_{ext} do not share a common node.

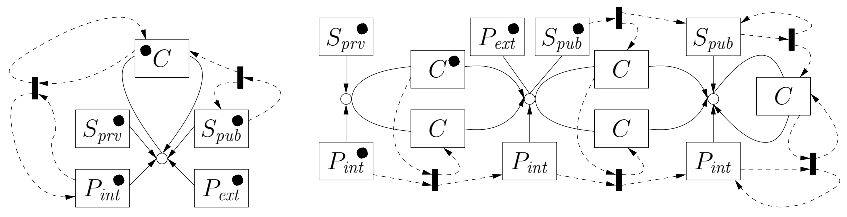

Fig. 2. The 0-covering $\mathcal{C}^0(\mathcal{S})$ and the 1-covering $\mathcal{C}^1(\mathcal{S})$ of grammar $\mathcal{S}$ in Fig. 1.

It would also be possible to extend the example by adding rules that allow movement of external processes and verify the same property. However, in this case the 1-covering would get larger and harder to draw. In [2] we have shown how to analyse a more complex GTS.

6 Conclusion

We have presented a technique for computing under- and over-approximations of the behaviour of graph transformation systems and we have identified suitable classes of properties of a GTS which can be inferred by analysing its approximations. We envision a scenario where a property of a given GTS can be checked by computing better and better approximations and verifying the property for each of them. Because of undecidability issues, this process might never terminate and it could also be costly from a complexity point of view, but with appropriate heuristics and fine-tuning of the technique, it is conceivable that several interesting properties for non-trivial GTSs can be verified in such a way.

In order to test the applicability of our theory we plan to implement the presented algorithm and to apply it to practical examples.

On the theoretical side, there are still several open problems. First, it would be interesting to classify logical formulae on graphs which are preserved and reflected by graph morphisms, via a kind of type system. This would enable us to extend the results of Section 5 to a logic in which one is able to reason specifically about graph transition systems (see also [6]). Additionally it would be necessary to detail how the verification of these formulae on Petri graphs can be reduced to the existing model-checking techniques for Petri nets.

Another relevant issue is the extension of the developed theory to GTSs having more general forms of rules. Particularly promising, in order to decrease the size of the approximations, is the case of GTSs where rules might have a non-discrete left-hand side. This extension would require to resort to contextual nets in order to represent the Petri net structure underlying a Petri graph.

An open and, as it seems, highly non-trivial question is the treatment of finite-state GTSs. It would be quite interesting to understand if for a given GTS with only finitely many reachable graphs (up to isomorphism), there is a way to construct—using folding and unfolding steps—a finite Petri graph which gives an exact representation of the original GTS, without any proper approximation. This would allow to reduce the analysis of finite-state GTSs to that of Petri nets.

Acknowledgements: The authors are grateful to Andrea Corradini for his insightful suggestions and to the anonymous referees for their comments on the submitted version of this paper.

References

1. P. Baldan, A. Corradini, and B. König. A static analysis technique for graph transformation systems. In *Proc. of CONCUR '01*, pages 381–395. Springer-Verlag, 2001. LNCS 2154.
2. P. Baldan, A. Corradini, and B. König. Static analysis of distributed systems with mobility specified by graph grammars—a case study. In *Proc. of IDPT '02 (World Conference on Integrated Design & Process Technology)*, 2002. to appear.
3. P. Baldan, A. Corradini, and U. Montanari. Concatenable graph processes: relating processes and derivation traces. In *Proc. of ICALP'98*, volume 1443 of *LNCS*, pages 283–295. Springer Verlag, 1998.
4. P. Baldan, A. Corradini, and U. Montanari. Unfolding and Event Structure Semantics for Graph Grammars. In W. Thomas, editor, *Proc. of FoSSaCS '99*, volume 1578 of *LNCS*, pages 73–89. Springer Verlag, 1999.
5. A. Corradini, U. Montanari, and F. Rossi. Graph processes. *Fundamenta Informaticae*, 26:241–265, 1996.
6. B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 5. World Scientific, 1997.
7. P. Cousot. Abstract interpretation. *ACM Computing Surveys*, 28(2), 1996.
8. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Formal Models and Semantics, Handbook of Theoretical Computer Science*, volume B, pages 243–320. Elsevier, 1990.

9. H. Ehrig. Introduction to the algebraic theory of graph grammars. In V. Claus, H. Ehrig, and G. Rozenberg, editors, *Proc. of the 1st International Workshop on Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *LNCS*, pages 1–69. Springer Verlag, 1979.
10. H. Ehrig, H.-J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.3: Concurrency, Parallellism, and Distribution.* World Scientific, 1999.
11. H. Ehrig, J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2: Concurrency, Parallelism and Distribution.* World Scientific, 1999.
12. J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23(2–3):151–195, 1994.
13. J. Esparza. Decidability of model-checking for infinite-state concurrent systems. *Acta Informatica*, 34:85–107, 1997.
14. T. Genet. Decidable approximations of sets of descendants and sets of normal forms. In T. Nipkow, editor, *Proc. of 9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *LNCS*, pages 151–165. Springer Verlag, 1998.
15. R.R. Howell, L.E. Rosier, and H.-C. Yen. A taxonomy of fairness and temporal logic problems for Petri nets. *Theoretical Computer Science*, 82:341–372, 1991.
16. P. Jančar. Decidability of a temporal logic problem for Petri nets. *Theoretical Computer Science*, 74:71–93, 1990.
17. N.D. Jones and F. Nielson. Abstract interpretation: a semantics-based tool for program analysis. In S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science, Vol. 4: Semantic Modelling*, pages 527–636. Oxford University Press, 1995.
18. M. Koch. *Integration of Graph Transformation and Temporal Logic for the Specification of Distributed Systems.* PhD thesis, Technische Universität Berlin, 2000.
19. B. König. A general framework for types in graph rewriting. In *Proc. of FST TCS 2000*, volume 1974 of *LNCS*, pages 373–384. Springer-Verlag, 2000.
20. C. Loiseaux, S. Graf, J. Sifakis, A. Bouajjani, and S. Bensalem. Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, 6:1–35, 1995.
21. P. Manolios and R.J. Trefler. Safety and liveness in branching time. In *Proc. of LICS '01*, 2001.
22. K.L. McMillan. *Symbolic Model Checking.* Kluwer, 1993.
23. J. Meseguer and U. Montanari. Petri nets are monoids. *Information and Computation*, 88:105–155, 1990.
24. A. Prasad Sistla. Safety, liveness and fairness in temporal logic. *Formal Aspects of Computing*, 6(5):495–512, 1994.
25. W. Reisig. *Petri Nets: An Introduction.* EACTS Monographs on Theoretical Computer Science. Springer Verlag, 1985.
26. L. Ribeiro. *Parallel Composition and Unfolding Semantics of Graph Grammars.* PhD thesis, Technische Universität Berlin, 1996.
27. G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations.* World Scientific, 1997.
28. G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations.* World Scientific, 1997.
29. V. Sassone. *On the Semantics of Petri Nets: Processes, Unfolding and Infinite Computations.* PhD thesis, University of Pisa - Department of Computer Science, 1994.