Towards Unfolding-Based Verification for Graph Transformation Systems

Paolo Baldan [1,2]

Dipartimento di Informatica
Università Ca' Foscari di Venezia
Italy

Andrea Corradini [1,3]

Dipartimento di Informatica
Università di Pisa
Italy

Barbara König [4]

Institut für Informatik
Technische Universität München
Germany

Abstract

The unfolding semantics of graph transformation systems can represent a basis for their formal verification. For general, possibly *infinite-state*, graph transformation systems one can construct finite under- and over- approximations of the (infinite) unfolding, with arbitrary accuracy. Such approximations can be used to check properties of a graph transformation system, like safety and liveness properties, expressed in suitable fragments of the μ-calculus. For *finite-state* graph transformation systems, a variant of McMillan's approach (originally developed for Petri nets) allows us to single out a finite under-approximation which is a so-called *complete prefix* of the unfolding, i.e., which provides an "exact" representation of the behaviour the original system as far as reachable states are concerned. Some problems related to the constructive definition of the prefix are discussed.

[1] Research supported by the MIUR Project COFIN 2001013518 CoMeta and by the FET-GC Projects IST-2001-32747 Agile.

[2] Email: baldan@dsi.unive.it

[3] Email: andrea@di.unipi.it

[4] Email: koenigb@in.tum.de

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

1 Introduction

Graph transformation systems (GTSs) [15] are recognized as a powerful specification formalism for concurrent and distributed systems [7], generalizing Petri nets. Along the years their truly concurrent behaviour has been deeply studied and a consolidated theory of concurrency is now available [15,7]. In particular, several semantics of Petri nets, like process and unfolding semantics, have been extended to GTSs (see, e.g., [6,14,2,3]). However, concerning automated verification, while several approaches exist for Petri nets, ranging from the calculus of invariants [13] to model checking based on finite complete prefixes [12], the rich literature on GTSs does not contain many contributions to the static analysis of such systems (see [10,11,9]).

In fact most of the mentioned semantics for GTSs define a (possibly concurrent) 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, hardly usable in a direct way for model-checking purposes. A common schema in the literature consists of considering an abstraction $\mathcal{A}$ of the concrete semantical model, providing a description of the behaviour of the system which is approximated, but still useful to check some properties of interest. Fixed a class $\mathcal{L}$ of properties of interest, the construction of the abstraction is typically guided by $\mathcal{L}$, in a way that given any property φ in $\mathcal{L}$, the validity of φ in abstraction $\mathcal{A}$ implies its validity in the original system. In some optimal cases, also the converse holds, i.e., the abstraction is "exact" for the properties in $\mathcal{L}$.

We focus on the unfolding semantics as a description of the behaviour 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. It is constructed [14,3] inductively beginning from the start graph and then applying at each step in all possible ways the rules, without deleting the left-hand sides, and recording each occurrence of a rule and each new graph item generated in the rewriting process.

The unfolding is infinite for any non-trivial GTS. Here, concentrating on GTSs where rules do not delete nodes, we suggest how one can define finite structures, approximating the full unfolding, where interesting classes of properties of the original system can be studied and verified. We remark that the impossibility of deleting nodes is not a severe limitation since the deletion of a node can be simulated by leaving the node isolated.

- For general, possibly *infinite-state* (hyper)graph transformation systems one can construct finite 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.

- For *finite-state* graph transformation systems a generalization of McMillan's approach (originally developed for Petri nets) allows us to single out a finite under-approximation which is a so-called *complete prefix* of the unfolding, providing an "exact" abstraction of the behaviour the original system as far as reachable states are concerned. Some problems related to the constructive definition of the prefix are discussed.

Some results are still in a preliminary shape, but they are promising in suggesting how the unfolding approach to the semantics of GTSs can represent a solid theoretical basis for a formal verification activity.

2 Approximations of Infinite-State GTSs

Let $\mathcal{G}$ be a general, possibly *infinite-state graph grammar*, i.e., GTS with a start hypergraph representing the initial state of the system. Here we consider basic graph grammars, without any distinction between terminal and non-terminal symbols and without any high-level control mechanism. As shown in [1,5] variations of the unfolding algorithm can be used to construct finite structures which can be seen as approximations of the full unfolding of the grammar, at a chosen level k of accuracy.

2.1 Under-approximations (k-truncations)

The unfolding of the 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 general, for infinite state systems, any truncation of the unfolding will be a proper 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.

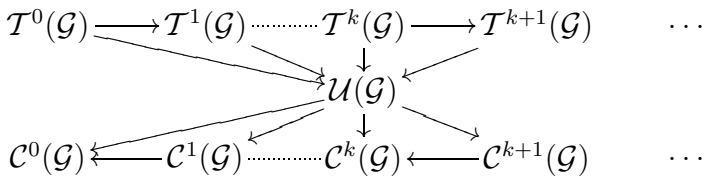
2.2 Over-approximations (k-coverings).

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 the papers [1,5] 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. The outcome of the algorithm is

not uniquely determined by the graph grammar, but changes according to the chosen level of accuracy: essentially 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 $\mathcal{C}^k(\mathcal{G})$ and 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 proved directly on the Petri net component of the coverings.

The unfolding is approximated by k-truncations and k-coverings with arbitrary high accuracy, a fact that is formalized by proving that both under- and over-approximations of the unfolding, converge to the full unfolding. In categorical terms we show that 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 $\mathcal{C}^k(\mathcal{G})$:



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.

3 Complete Prefix for Finite-State GTSs

For Petri nets and for other formalisms endowed with an unfolding semantics, it has been shown that, when the system is finite-state, it is possible to identify a finite initial part of the unfolding, called *finite complete prefix* [12,8], which provides an exact approximation of the system as far as reachability properties are concerned. A complete prefix represents all and only the markings reachable in the original Petri net, in the sense that the reachable markings corresponds exactly to the concurrent subsets of places in the finite prefix.

To construct the finite prefix the idea consists of unfolding the given Petri

net avoiding to insert "useless" transitions, where "useless" means transitions which do not contribute to generate new markings. More precisely the construction stops at the so-called *cut-off* events, which, roughly speaking, are transitions which produce the same marking as other transitions but with a larger causal history (i.e., in a larger number of steps).

Finite prefixes have been used for the verification of properties like absence of deadlocks or hazard-freedom in logic circuits [12,16] and for the model checking of a simple but quite powerful branching temporal logic over Petri nets [8], where net properties like reachability, mutual exclusion, concurrency, liveness, cyclicity can be expressed.

We suggest how the complete prefix approach can be generalized to our hypergraph transformation systems. More precisely, using the notation introduced in the previous section, we prove that if $\mathcal{G}$ is a finite-state graph grammar then there exists a level of accuracy k such that $\mathcal{T}^k(\mathcal{G})$ is a complete prefix, i.e., $\mathcal{T}^k(\mathcal{G})$ provides an exact approximation of the behaviour of $\mathcal{G}$ as far as reachable states are concerned.

Notably, in the case of graph transformation systems, the possibility of performing "contextual" rewritings, i.e., of preserving part of the state in a rewriting step, leads to a form of asymmetric conflict (or weak causality) between events. Hence differently from what happens for (ordinary) Petri nets, in graph grammar computations an event does not have a uniquely determined causal history (consisting of the set of its causes), but it has instead a set of different local histories [4,3]. The definition of cut-off event must be updated consequently: an event is called a *generalized cut-off* if for *any* of its histories there exists another event with a smaller local history, producing the same final state.

Exploiting this definition of generalized cut-off, we are able to prove the *existence* of a finite complete prefix for any finite-state graph grammar. Unfortunately, for general hypergraph transformation systems, the proof is non-constructive. Therefore it is still an open problem to devise an algorithm for constructing such a finite complete prefix, the main difficulty being the non-monotonicity of the notion of generalized cut-off.

References

[1] P. Baldan, A. Corradini, and B. König. A static analysis technique for graph transformation systems. In K.G. Larsen and M. Nielsen, editors, *Proceedings of CONCUR 2001*, volume 2154 of *LNCS*, pages 381–395. Springer Verlag, 2001.

[2] P. Baldan, A. Corradini, and U. Montanari. Concatenable graph processes: relating processes and derivation traces. In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 283–295. Springer Verlag, 1998.

[3] P. Baldan, A. Corradini, and U. Montanari. Unfolding and Event Structure

Semantics for Graph Grammars. In W. Thomas, editor, *Proceedings of FoSSaCS '99*, volume 1578 of *LNCS*, pages 73–89. Springer Verlag, 1999.

[4] P. Baldan, A. Corradini, and U. Montanari. Contextual Petri nets, asymmetric event structures and processes. *Information and Computation*, 171(1):1–49, 2001.

[5] P. Baldan and B. König. Approximating the behaviour of graph transformation systems. In A. Corradini, H. Ehrig, H.-J. Kreowski, and G. Rozemberg, editors, *Proceedings of ICGT'02*, volume 2505 of *LNCS*, pages 14–30. Springer Verlag, 2002.

[6] A. Corradini, U. Montanari, and F. Rossi. Graph processes. *Fundamenta Informaticae*, 26:241–265, 1996.

[7] H. Ehrig, J. Kreowski, U. Montanari, and G. Rozenberg, editors. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 3: Concurrency, Parallelism and Distribution*. World Scientific, 1999.

[8] J. Esparza. Model checking using net unfoldings. *Science of Computer Programming*, 23(2–3):151–195, 1994.

[9] R. Heckel. Compositional verification of reactive systems specified by graph transformation. In E. Astesiano, editor, *Porceedings of FASE'98*, volume 1382 of *LNCS*, pages 138–153. Springer Verlag, 1998.

[10] M. Koch. *Integration of Graph Transformation and Temporal Logic for the Specification of Distributed Systems*. PhD thesis, Technische Universität Berlin, 2000.

[11] 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.

[12] K.L. McMillan. *Symbolic Model Checking*. Kluwer, 1993.

[13] W. Reisig. *Petri Nets: An Introduction*. EACTS Monographs on Theoretical Computer Science. Springer Verlag, 1985.

[14] L. Ribeiro. *Parallel Composition and Unfolding Semantics of Graph Grammars*. PhD thesis, Technische Universität Berlin, 1996.

[15] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*. World Scientific, 1997.

[16] W. Vogler, A. Semenov, and A. Yakovlev. Unfolding and finite prefix for nets with read arcs. In *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 501–516. Springer-Verlag, 1998.