Applying the Graph Minor Theorem to the Verification of Graph Transformation Systems*

Salil Joshi[1] and Barbara König[2]

[1] Indian Institute of Technology, Delhi, India
[2] Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, Germany

Abstract. We show how to view certain subclasses of (single-pushout) graph transformation systems as well-structured transition systems, which leads to decidability of the covering problem via a backward analysis. As the well-quasi order required for a well-structured transition system we use the graph minor ordering. We give an explicit construction of the backward step and apply our theory in order to show the correctness of a leader election protocol.

1 Introduction

In a series of seminal papers Robertson and Seymour have shown that graphs are well-quasi-ordered with respect to the minor ordering [7,8]: in any (infinite) sequence of graphs $G_0, G_1, G_2, \ldots$ there are always two indices $i < j$ such that G_i is a minor of G_j. This means that G_i can be obtained from G_j by deleting and contracting edges and by deleting isolated nodes.

The theorem has far-reaching consequences. It guarantees that every set of graphs that is upward-closed with respect to the minor ordering can be represented by a finite number of minimal graphs. Similarly, any downward-closed set of graphs (e.g., planar graphs, forests, graphs embeddable in a torus) can be characterized by a finite set of forbidden minors. A well-known special case are (undirected) planar graphs which are characterized by two forbidden minors: the complete graph with five nodes (K_5) and the complete bipartite graph with six nodes ($K_{3,3}$), a fact which is known as Kuratowski's theorem.

Well-quasi-orders (wqo's) also play a fundamental role in the analysis of a class of (infinite-state) transition systems, so called well-structured transition systems (WSTS) [4]. States in a WSTS are well-quasi-ordered and the standard analysis method shows whether some state in an upward-closed set is reachable from an initial state by performing backward analysis. The well-quasi-ordering guarantees that upward-closed sets are finitely representable, that the set of predecessors is also upward-closed and that the technique terminates after finitely many steps.

One important example for WSTS are Petri net transition system, where a marking m_1 is considered larger than or equal to m_2 if it contains at least as

* Research partially supported by the DFG project SANDS.

A. Gupta and S. Malik (Eds.): CAV 2008, LNCS 5123, pp. 214–226, 2008.

many tokens in every place. Other examples are string rewrite systems, basic process algebra and communicating finite state machines. A transition system that can not be naturally viewed as a WSTS can often be turned into one by introducing some notion of "lossiness". For instance an unreliable channel may lose messages and a suitable wqo considers the content c_1 of a channel as greater than c_2 if c_2 can be obtained from c_1 by dropping some messages.

The graph minor ordering fits well with this intuition of "lossiness" and seems to be applicable to networks where edges (connections or processes) may disappear—possibly due to faults—and where edges can be contracted. The latter phenomenon appears if a process leaves a network by connecting its predecessor and successor, something which typically happens in rings.

Here we show how to view certain graph transformation systems (GTS) as WSTS with respect to the minor ordering. GTS are an intuitive formalism, well-suited to model concurrent and distributed systems. In general GTS are Turing-complete and due to undecidability issues it is hard to imagine a useful wqo for the general case. However, if the GTS exhibits features as described above it can be successfully verified.

GTS are typically defined by means of category theory, which makes the definition of rewriting steps less tedious. Graph rewriting is defined via pushouts in a suitable category of graph morphisms and in the rest of this paper we will exploit certain well-known properties of pushouts. The relation of a graph G to its minor H can be represented by a partial graph morphism with specific properties. Since the theory requires the handling of partial morphisms, we have decided to work in the single-pushout approach (SPO) which uses partial morphisms [5,3].

The paper is organized as follows: Section 2 introduces the basic definitions. In Section 3 we consider classes of GTS that can be seen as WSTS, and introduce the techniques for their analysis. In Section 4 we will look at a leader election protocol and show how the analysis method works in practice.

2 Preliminaries

Here we introduce some of the basic notions needed in the paper, especially well-quasi-orders, well-structured transition systems, graph transformation systems and minors.

2.1 Well-Quasi-Order

Definition 1 (wqo). *A* well-quasi-order *(wqo) is any quasi-ordering*[1] $\leq$ *(over some set X) such that, for any infinite sequence $x_0, x_1, x_2, \ldots$ in X, there exist indices $i < j$ with $x_i \leq x_j$.*

An upward-closed set *is any set $I \subseteq X$ such that $y \geq x$ and $x \in I$ entail $y \in I$. A* downward-closed set *can be analogously defined.*

For an element $x \in I$, we define $\uparrow x = \{y \mid y \geq x\}$. Then, a basis *of an upward-closed set I is a set I^b such that $I = \bigcup_{x \in I^b} \uparrow x$.*

[1] Note that a quasi-order is the same as a preorder.

Lemma 2

1. *If $\leq$ is a well-quasi-ordering then any upward-closed I has a finite basis.*
2. *If $\leq$ is a wqo and $I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots$ is an infinite increasing sequence of upward-closed sets, then there exists an index $k \in \mathbb{N}$ such that $I_k = I_{k+1} = I_{k+2} = \ldots$*

2.2 Well-Structured Transition Systems

Definition 3 (WSTS). *A* well-structured transition system *(WSTS) is a transition system $T = (S, \Rightarrow, \leq)$, where S is a set of states and $\Rightarrow \subseteq S \times S$, such that the following conditions hold:*

1. ***Well quasi ordering:*** *$\leq$ is a well-quasi-ordering on S.*
2. ***Compatibility:*** *For all $s_1 \leq t_1$ and a transition $s_1 \Rightarrow s_2$, there exists a sequence $t_1 \Rightarrow^* t_2$ of transitions such that $s_2 \leq t_2$.*

$$\begin{array}{ccc} t_1 & \Longrightarrow^* & t_2 \\ \vee| & & \vee| \\ s_1 & \Longrightarrow & s_2 \end{array}$$

Given a set $I \subseteq S$ of states we denote by $Pred(I)$ the set of direct predecessors of I, i.e., $Pred(I) = \{s \in S \mid \exists s' \in I : s \Rightarrow s'\}$. Furthermore $Pred^*(I)$ is the set of all predecessors.

Let $(S, \Rightarrow, \leq)$ be a WSTS. Consider a set of states $I \subseteq S$. Backward reachability analysis involves the computation of $Pred^*(I)$ as the limit of the sequence $I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots$ where $I_0 = I$ and $I_{n+1} = I_n \cup Pred(I_n)$. However, in general this may not terminate. For WSTS, if I is upward-closed then it can be shown that $Pred^*(I)$ is also upward-closed (compatibility condition) and that termination is guaranteed (Lemma 2).

Definition 4 (Effective pred-basis). *A WSTS has an* effective pred-basis *if there exists an algorithm accepting any state $s \in S$ and returning $pb(s)$, a finite basis of $\uparrow Pred(\uparrow s)$.*

Now assume that T is a WSTS with effective pred-basis. Pick a finite basis I^b of I and define a sequence $K_0, K_1, K_2, \ldots$ of sets with $K_0 = I^b$ and $K_{n+1} = K_n \cup pb(K_n)$. Let m be the first index such that $\uparrow K_m = \uparrow K_{m+1}$. Such an m must exist by Lemma 2 and we have $\uparrow K_m = Pred^*(I)$. Finally, note that due to Lemma 2 every set K_n can be represented by a finite basis.

The *covering problem* is to decide, given two states s and t, whether starting from a state s it is possible to cover t, i.e. to reach a state t' such that $t' \geq t$. From the previous argument follows the decidability of the covering problem.

Theorem 5 (Covering problem). *The covering problem is decidable for a WSTS with an effective pred-basis and a decidable wqo $\leq$.*

Thus, if T is a WSTS and the "error states" can be represented as an upward-closed set I, then it is decidable whether any element of I is reachable from the start state.

2.3 Graphs and Graph Transformation

Definition 6 (Hypergraph). *Let Λ be a finite set of labels. A* (Λ-)hypergraph *is a tuple (V_G, E_G, c_G, l_G) where V_G is a finite set of nodes, E_G is a finite set of edges, $c_G\colon E_G \to V_G^*$ is a connection function and $l_G\colon E_G \to \Lambda$ is the labelling function for edges.*

Directed labelled graphs are a special case of hypergraphs where every sequence $c_G(e)$ is of length two.

Definition 7 (Partial hypergraph morphism). *Let G, G' be (Λ-)hypergraphs. A* partial hypergraph morphism *(or simply* morphism*) $\varphi\colon G \rightharpoonup G'$ consists of a pair of partial functions $(\varphi_V : V_G \rightharpoonup V_{G'}, \varphi_E : E_G \rightharpoonup E_{G'})$ such that for every $e \in E_G$ it holds that $l_G(e) = l_{G'}(\varphi_E(e))$ and $\varphi_V(c_G(e)) = c_{G'}(\varphi_E(e))$ whenever $\varphi_E(e)$ is defined. Furthermore if a morphism is defined on an edge, it must be defined on all nodes adjacent to it. (This condition need not hold in the other direction.)*

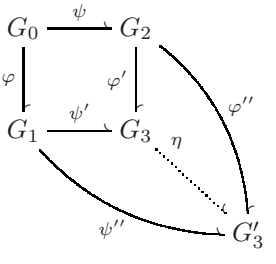
Total morphisms *are denoted by an arrow of the form $\to$.*

In the following we will drop the subscripts and write φ instead of φ_V and φ_E.

Gluing of graphs along a common subgraph is done via pushouts in the category of partial graph morphisms.

Definition 8 (Pushout)

Let $\varphi\colon G_0 \rightharpoonup G_1$ and $\psi\colon G_0 \rightharpoonup G_2$ be two partial graph morphisms. The pushout *of φ and ψ consists of a graph G_3 and two graph morphisms $\psi'\colon G_1 \rightharpoonup G_3$, $\varphi'\colon G_2 \rightharpoonup G_3$ such that $\psi' \circ \varphi = \varphi' \circ \psi$ and for every other pair of morphisms $\psi''\colon G_1 \rightharpoonup G_3'$, $\varphi''\colon G_2 \rightharpoonup G_3'$ such that $\psi'' \circ \varphi = \varphi'' \circ \psi$ there exists a unique morphism $\eta\colon G_3 \rightharpoonup G_3'$ with $\eta \circ \psi' = \psi''$ and $\eta \circ \varphi' = \varphi''$.*



It is known that pushouts of partial graph morphisms always exist, that they are unique up to isomorphism and that they can be constructed as follows. The intuition behind the construction is that G_1, G_2 are glued together along a common interface G_0 and that an element is deleted if it is deleted by either φ or ψ.

Proposition 9 (Construction of pushouts). *Let $\varphi\colon G_0 \rightharpoonup G_1$, $\psi\colon G_0 \rightharpoonup G_2$ be partial hypergraph morphisms. Furthermore let $\equiv_V$ be the smallest equivalence on $V_{G_1} \cup V_{G_2}$ and $\equiv_E$ the smallest equivalence on $E_{G_1} \cup E_{G_2}$ such that $\varphi(x) \equiv \psi(x)$ for every element x of G_0.*

An equivalence class of nodes is called valid *if it does not contain the image of a node x for which $\varphi(x)$ or $\psi(x)$ are undefined. Similarly a class of edges is* valid *if the analogous condition holds and furthermore all nodes adjacent to these edges are contained in valid equivalence classes.*

Then the pushout G_3 of φ and ψ consists of all valid equivalence classes $[x]_\equiv$ as nodes and edges, where $l_{G_3}([e]_\equiv) = l_{G_i}(e)$ and $c_{G_3}([e]_\equiv) = [v_1]_\equiv \dots [v_k]_\equiv$ if $e \in E_{G_i}$ and $c_{G_i}(e) = v_1 \dots v_k$.

It can be seen that the pushout of two total morphisms (in the category of partial morphisms) always results in two total morphisms. Furthermore it is equal to their pushout in the category of total morphisms. However φ total and ψ partial does not necessarily imply that φ' is total. This is due to so-called *deletion/preservation conflicts* where two elements x_0, x_0' of G_0 are mapped to the same element of G_1, i.e., $\varphi(x_0) = \varphi(x_0')$, while $\psi(x_0)$ is defined, whereas $\psi(x_0')$ is undefined. The construction above suggests that then $\varphi'(\psi(x_0))$ must be undefined, i.e., φ' is not total. If no such elements x_0, x_0' can be found, then φ is said to be *conflict-free* with respect to ψ and in this case φ' is always total.

Definition 10 (Graph rewriting). *A* rewriting rule *is a partial morphism $r\colon L \rightharpoonup R$, where L is called left-hand side and R right-hand side.*

A match *(of r) is a total morphism $m\colon L \to G$ which is conflict-free wrt. r.*

Given a rule and a match, a rewriting step *or an application of the rule to the graph G, resulting in H, is a pushout diagram as shown in Fig. 1 on the left. In this case we write $G \Rightarrow H$.*

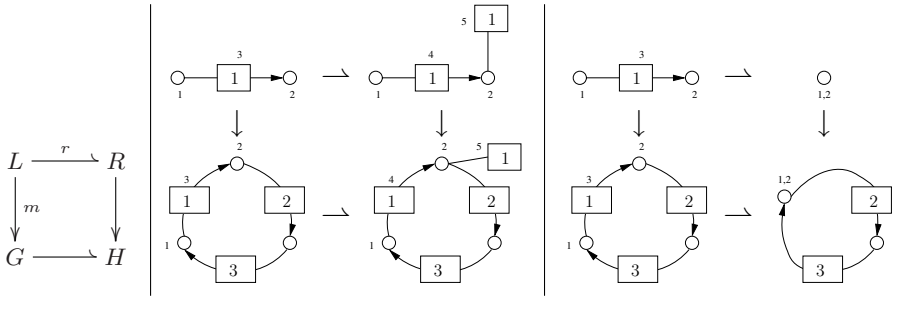


Fig. 1. Single-pushout graph rewriting (pushout diagram and example rewriting steps)

Intuitively, we can think of this as follows: L is a subgraph of G, all items of L whose image is undefined under r are deleted, the new items of R are added and connected as specified by r. Note that whenever a node is deleted, all adjacent edges will be deleted as well.

Fig. 1 shows two examples for graph rewriting steps. In the middle pushout a binary hyperedge generates another (unary) hyperedge, whereas in the right pushout an edge is contracted. The way in which the morphisms map nodes and edges is indicated by the small numbers next to the edges. These specific rewriting rules will also play a role in our application (see Section 4).

In the context of this paper a *graph transformation system (GTS)* consists of a finite set $\mathcal{R}$ of rewriting rules. Sometimes we will fix an *initial graph* or *start graph*.

2.4 Minors and Minor Morphisms

We will now review the notion of a graph minor.

Definition 11 (Minor). *A graph $\hat{G}$ is a* minor *of a graph G, if $\hat{G}$ can be obtained from G by (repeatedly) performing the following operations on G:*

1. *Deletion of an edge.*
2. *Contraction of an edge, thereby merging all nodes adjacent to the edge.*
3. *Deletion of an isolated node.*

The *Robertson-Seymour Theorem* [7] says that the minor order is a well-quasi-order. In fact, this theorem is true even if the edges and vertices of the graphs are labelled from a well-quasi-ordered set, and also for hypergraphs and directed graphs (see [8]).

Now, if we could show that a GTS satisfies the compatibility condition of Definition 3 (with respect to the minor ordering), we could analyze it using the theory of WSTS. But before we characterize such GTS we first need the definition of minor morphisms and their properties. A *minor morphism* is a partial morphism that identifies a minor of a graph.

Definition 12 (Minor morphism). *A partial morphism $\mu : G \rightharpoonup \hat{G}$ is a minor morphism (written $\mu : G \mapsto \hat{G}$) if*

1. *it is surjective,*
2. *it is injective on edges* and
3. *whenever $\mu(v) = \mu(w) = z$ for some $v, w \in V_G$ and $z \in V_{\hat{G}}$, there exists a path between v and w in G. If e is an edge on this path then $\mu(e)$ is undefined, and all nodes in $c_G(e)$ are mapped to z.*

In [8] a different way to characterize minors is proposed: a function, going in the opposite direction, mapping nodes of $\hat{G}$ to subgraphs of G. This however can not be seen as a morphism in the sense of Definition 7 and we would have problems integrating it properly into the theory of graph rewriting.

One can show the following facts about minor morphisms.

Lemma 13. *$\hat{G}$ is a minor of G iff there exists a minor morphism $\mu : G \mapsto \hat{G}$.*

Lemma 14. *Pushouts preserve minor morphisms in the following sense: If $f : G_0 \mapsto G_1$ is a minor morphism and $g : G_0 \to G_2$ is total, then the morphism f' in the pushout diagram below is a minor morphism.*

$$\begin{array}{ccc} G_0 & \overset{f}{\longmapsto} & G_1 \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle g'} \\ G_2 & \overset{f'}{\longmapsto} & G_3 \end{array}$$

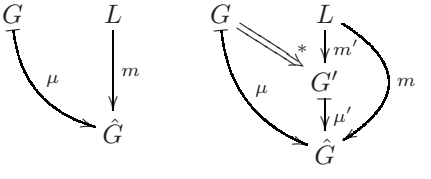
3 GTS as WSTS!

As observed earlier, a GTS can be seen as a WSTS with the minor relation as the well-quasi-ordering, provided the GTS satisfies the compatibility condition introduced in Definition 3.

3.1 Characterization

We will first give a sufficient condition that allows us to view a GTS as a WSTS. Note that the fundamental problem is that whenever a minor of G contains a left-hand side, then G might contain a "disconnected" copy of the left-hand side.

Proposition 15 (GTS as WSTS). *Let $\mathcal{R}$ be a GTS that satisfies the following condition: For every rule $(r\colon L \rightharpoonup R) \in \mathcal{R}$, every minor morphism $\mu\colon G \mapsto \hat{G}$ and every match $m\colon L \to \hat{G}$ (see diagram on the left) there exists a graph G' such that $G \Rightarrow^* G'$, there is a minor morphism $\mu'\colon G' \mapsto \hat{G}$ and there exists a match $m'\colon L \to G'$ such that $m = \mu' \circ m'$ (see commuting diagram below on the right). Then $\mathcal{R}$ is a WSTS.*



With this characterization we can now identify suitable types of GTS that are WSTS:

- Context-free graph grammars, where the left-hand side of every rule consists of a single hyperedge. Here G must always contain a match of L that makes the above diagram commute and no intermediate graph G' is needed.
- GTS where the left-hand sides of the rules consist of disconnected edges. The argument is analogous to the case above.
- Any arbitrary GTS can be transformed into a WSTS with the addition of an edge contraction rule for every edge label. Now, if $\hat{G}$ contains a subgraph which is isomorphic to a left-hand side, the pre-image of this subgraph under μ is present in G, but it might possibly be disconnected. The minor morphism μ makes the elements of L adjacent by contracting paths and the same can be done by applying the additional edge contraction rules.

3.2 Backward Analysis

Let $\mathcal{R}$ be a set of graph transformation rules which satisfies the compatibility condition. Now we consider the question of performing a backward reachability analysis on $\mathcal{R}$ which requires a method for computing an effective pred-basis $pb(S)$ for a given graph S.

Our method will involve the backwards application of an SPO rewriting rule. This requires the completion of a diagram of the form $L \rightharpoonup R \to H$ by a graph G and morphisms $L \to G \rightharpoonup H$ such that the square is a pushout. Then G is a so-called *pushout complement*. Pushout complements are well-studied for total morphisms since they are an essential ingredient in double-pushout rewriting. For partial morphisms they have been studied to a lesser extent.

We will first demonstrate some issues that can arise with pushout complements: for instance, the two total morphisms $L \rightharpoonup R \to H$ shown in Fig. 2 (left) (edges and nodes are unlabelled, morphisms are indicated by numbers $1, 2$) have five different pushout complements. Note also that each pair of total morphisms has only finitely many pushout complements (up to isomorphism).

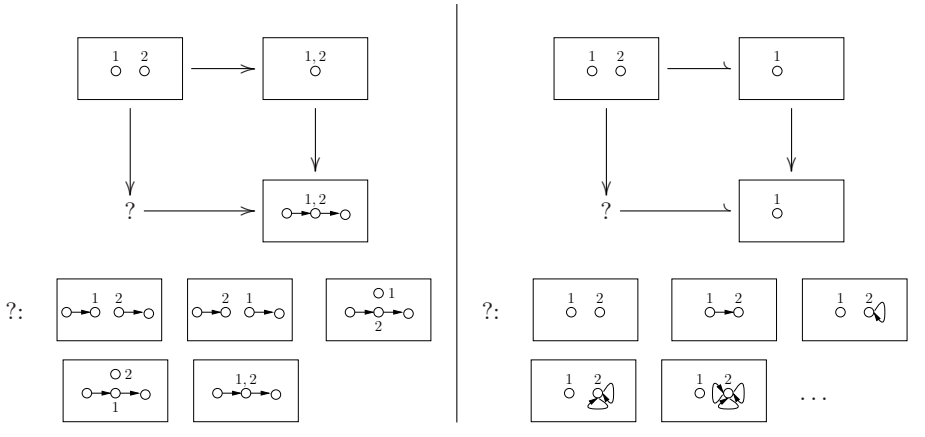


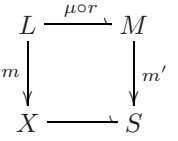
Fig. 2. Left: Two total morphisms with five pushout complements. Right: A partial and a total morphism with infinitely many pushout complements.

While the existence of multiple pushout complements is a feature that will be needed to determine the pred-basis, the situation for partial morphisms is more involved. Consider the diagram in Fig. 2 (right) where the morphism from L to R is partial. Here we have infinitely many pushout complements. Note however that the first graph is a minor of all other pushout complements. This suggests that only the computation of minimal pushout complements is needed.

Now we will give a high-level description of the procedure for computing $pb(S)$ for a given graph S. A more detailed account will be given in Section 3.3 where we will also argue that the procedure is indeed effective.

1. For each rule $(r : L \rightharpoonup R) \in \mathcal{R}$, let $\mathcal{M}_R$ be the (finite) set of all minor morphisms with source R.
2. For each $(\mu\colon R \mapsto M) \in \mathcal{M}_R$ consider the rule $\mu \circ r\colon L \rightharpoonup M$.
3. For each total match $m'\colon M \to S$ compute all minimal[2] pushout complements X such that $m\colon L \to X$ below is total and conflict-free wrt. r.

[2] "Minimimal" means "minimal wrt. the well-quasi ordering $\leq$".



4. The set $pb(S)$ contains all graphs X obtained in this way.

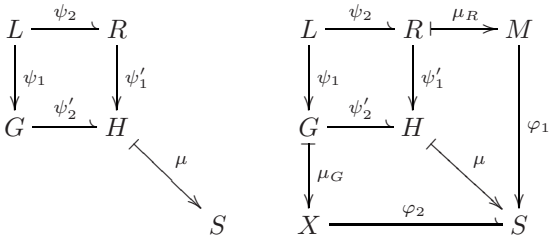
That is, we use all minors of R as right-hand sides for the backward step. This is needed since S represents an upward-closed set and not all items of R must be present in S itself. We can now show the correctness of the procedure $pb(S)$, where the proof depends crucially on Lemma 14.

Theorem 16. *The procedure $pb(S)$ computes a finite subset of $Pred(\uparrow S)$.*

In order to prove that $pb(s)$ generates every member of the pred-basis, we first prove a general result in the category of graphs and partial morphisms.

Lemma 17. *Let $\psi_1 : L \to G$ be total and conflict-free wrt. ψ_2. If the diagram below on the left is a pushout and $\mu : H \rightarrowtail S$ a minor morphism, then there exist minors M and X of R and G respectively, such that*

1. *the diagram below on the right commutes and the outer square is a pushout.*
2. *the morphisms $\mu_G \circ \psi_1 : L \to X$ and $\varphi_1 : M \to S$ are total and $\mu_G \circ \psi_1$ is conflict-free wrt. ψ_2.*



The lemma above says that whenever S is a minor of H and G is a predecessor of H, then we can make a backwards step for S and obtain X, a minor of G. Using this lemma we can now state the completeness of the procedure $pb(S)$.

Theorem 18. *The set generated by $pb(S)$ is a pred-basis of S.*

3.3 Computing Minimal Pushout Complements

Now we consider the question of how to construct pushout complements when some (but not all) of the morphisms involved may be partial. Hence consider a diagram $L \overset{\varphi}{\rightharpoonup} \tilde{L} \to \tilde{X}$. The idea is to split $L \overset{\varphi}{\rightharpoonup} \tilde{L} = L \rightharpoonup dom(\varphi) \to \tilde{L}$ where $dom(\varphi) \to \tilde{L}$ is total and $L \rightharpoonup dom(\varphi)$ is an inverse injection, i.e., a morphism which is injective, surjective, but not necessarily total. Now the task of computing pushout complements can be divided into two subtasks.

Lemma 19. *Let L and $\tilde{L}$ be graphs, $\varphi_1 : L \rightharpoonup \tilde{L}$ be an inverse injection, and $\psi_2 : \tilde{L} \to \tilde{X}$ be a total morphism. Now construct a specific pushout complement X' with morphisms $\psi_1' \colon L \to X'$, $\varphi_2' \colon X' \rightharpoonup \tilde{X}$ as follows:*

1. *Take a copy of the graph $\tilde{X}$, and let ψ_1' be $\psi_2 \circ \varphi_1$. The morphism φ_2' is the identity.*
2. *Let Y be the set of elements of L the image of which is undefined under φ_1. Add a copy of Y to this copy of $\tilde{X}$, and extend ψ_1' by mapping Y into this set. Furthermore φ_2' is undefined on all elements of the copy of Y.*
3. *Now merge these new elements (originally contained in Y) in all possible combinations, i.e., factor through all appropriate[3] equivalences. The morphisms ψ_1' and φ_2' are modified accordingly.*

The set of graphs obtained in this way is denoted by $\mathcal{P}$. Each element X' of $\mathcal{P}$ is a pushout complement of φ_1, ψ_2 and the corresponding morphisms $\psi_1' \colon L \to X'$ are total. Any other pushout complement X where $\psi_1 : L \to X$ is total (see diagram on the right) has some graph $X' \in \mathcal{P}$ as a minor.

$$\begin{array}{ccc} L & \xrightarrow{\varphi_1} & \tilde{L} \\ \downarrow{\scriptstyle \psi_1} & & \downarrow{\scriptstyle \psi_2} \\ X & \xrightarrow{\varphi_2} & \tilde{X} \end{array}$$

Finally, if $\psi_1 \colon L \to X$ is conflict-free wrt. to a rule $r : L \rightharpoonup R$, then there exists a pushout complement $X' \in \mathcal{P}$ with $\psi_1' \colon L \to X'$ conflict-free wrt. r, such that $X' \leq X$.

In order to do backwards application of rules in order to obtain $pb(s)$, we construct pushout complements (with total conflict-free matches) as follows:

Proposition 20. *Let $r \colon L \rightharpoonup R$ be a fixed rule. Furthermore let L, M and S be graphs, with a partial morphism $\varphi_1 : L \rightharpoonup M$ and a total morphism $\psi_2 : M \to S$. Then, if we apply the following procedure we only construct pushout complements X' of φ_1, ψ_2 and any other pushout complement X (with $\psi_1 \colon L \to X$ where ψ_1 is total and conflict-free wrt. r) has one of them as a minor.*

1. *Split φ_1 into two morphisms as follows: let $\varphi_1' : L \rightharpoonup dom(\varphi_1)$ be an inverse injection and let $\varphi_1'' : dom(\varphi_1) \to M$ be total.*
2. *Now consider the total morphisms $\varphi_1'' : dom(\varphi_1) \to M$, and $\psi_2 : M \to S$. Construct all their pushout complements as usual for total morphisms.*[4]
3. *Let $\tilde{X}$ be any such pushout complement with $\eta \colon dom(\varphi_1) \to \tilde{X}$.*
4. *For φ_1', η use the construction of Lemma 19 in order to obtain the minimal pushout complements X' (with total and conflict-free ψ_1').*
5. *Finally, from all such pushout complements X' take the minimal ones.*

The situation is depicted in the diagram below.

$$\begin{array}{ccccc} L & \xrightarrow{\varphi_1'} & dom(\varphi_1) & \xrightarrow{\varphi_1''} & M \\ \downarrow{\scriptstyle \psi_1'} & & \downarrow{\scriptstyle \eta} & & \downarrow{\scriptstyle \psi_2} \\ X' & \longrightarrow & \tilde{X} & \longrightarrow & S \end{array}$$

[3] Here "appropriate" means that whenever two edges are in the equivalence relation, all their adjacent nodes must be pairwise equivalent.

[4] We do not describe this construction here, but it is well-known that there are only finitely many such pushout complements and that they can be constructed effectively.

4 Example: Leader Election

As an example, we shall apply this technique to a typical leader election protocol, to verify its correctness. The rules for this leader election protocol are shown in Fig. 3. We start with a ring containing processes, each with a unique natural number as ID. These processes can generate messages containing their ID, which are forwarded whenever the ID of the message is smaller than the ID of the process which receives it. A process becomes the leader if it receives a message containing its own ID. Non-leader processes may also choose to leave the system at any time, connecting its predecessor and successor. We will prove that such a system can never create two leaders in the ring.

It can be seen that these rules satisfy the compatibility condition. The rule for edge contraction can be interpreted as a process leaving the system. Note that we do not need to add a rule for contracting messages (since messages are unary hyperedges), or for edge deletion in order to ensure compatibility.

All forbidden minors (which we computed manually) are shown in Fig. 4. We start with the first of these as the error state, and performing the backward analysis we obtain the rest of the forbidden minors. We consider natural numbers up to a certain bound, in order to keep the label and rule sets finite. Here, i, j or k as a label indicates "any number" (except where a constraint is indicated). Thus, the entire process has been fully parametrized, so that these forbidden minors are valid for a start graph with an arbitrarily large number of processes in the ring. Since the given start graph does not have any of these forbidden graphs as a minor, we can conclude that the leader election protocol is correct, i.e., it can never create two leaders.

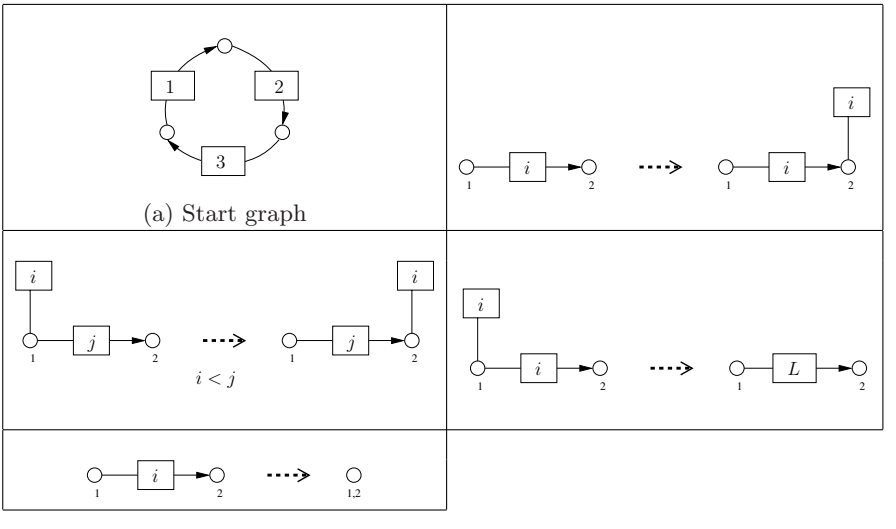


Fig. 3. Leader election (start graph and rewriting rules)

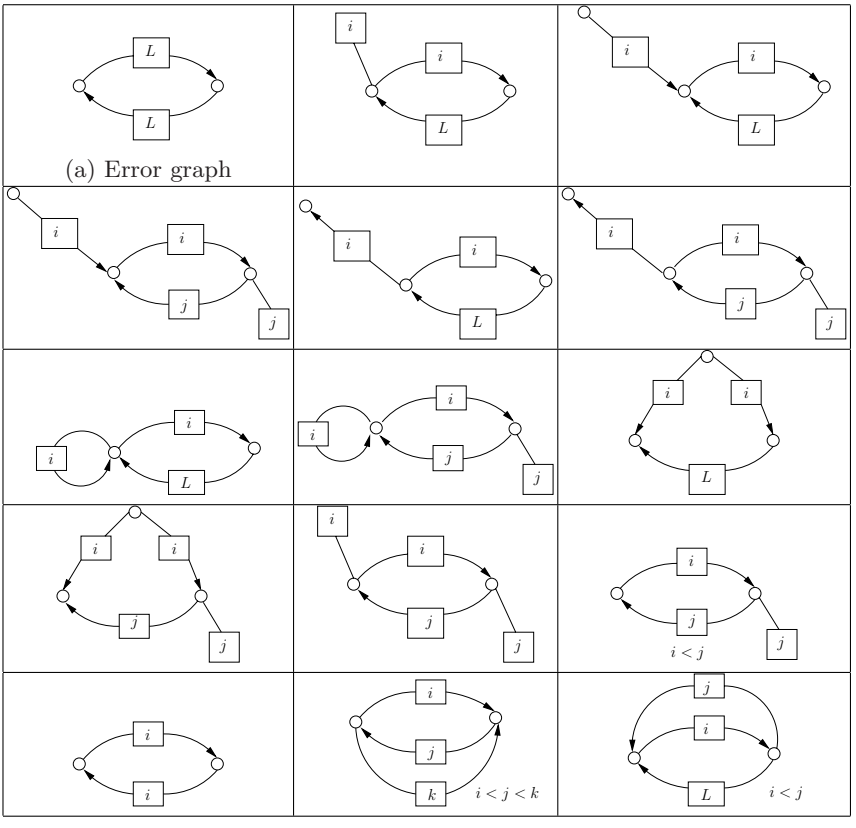


Fig. 4. Leader election (forbidden minors)

Note that since our technique can handle infinite state spaces, we could use the expressive power of graph transformation to extend the example in such a way that the ring is extended by new processes during runtime.

5 Conclusion

We have shown how to view subclasses of graph transformation systems as WSTS which gives us a decision algorithm for the covering problem. Currently we are working on an implementation which will help us to get a better insight into efficiency issues. Specifically it will help us to answer how many backward steps usually have to be taken and how many forbidden minors are generated. Although the worst case behaviour of this technique will certainly be bad, it might be feasible for many practical applications. We are also working on a more extended case study involving a termination detection protocol.

Another issue is the treatment of negative application conditions that have so far posed many problems in the analysis of GTSs. As already observed in [9]

backward techniques seem to have fewer problems with negative application conditions than forward techniques which have so far mainly been studied. We also believe that such application conditions can be integrated with our technique.

Additional future work will be the investigation of partial order techniques (as in [1]) and the combination with (approximative) forward techniques (as described in [2,6]) in order to eliminate states which are not reachable from the start graph early on. In addition we work on a related technique which allows to show whether certain invariants (represented by forbidden minors) are preserved by graph transformation rules.

Acknowledgements. We would like to thank Javier Esparza for his suggestion to explore the relation between WSTS and graph transformation.

References

1. Abdulla, P.A., Jonsson, B., Kindahl, M., Peled, D.: A general approach to partial order reductions in symbolic verification. In: Y. Vardi, M. (ed.) CAV 1998. LNCS, vol. 1427, pp. 379–390. Springer, Heidelberg (1998)
2. Baldan, P., Corradini, A., König, B.: A static analysis technique for graph transformation systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 381–395. Springer, Heidelberg (2001)
3. Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., Corradini, A.: Algebraic approaches to graph transformation—part II: Single pushout approach and comparison with double pushout approach. In: Rozenberg, G. (ed.) Handbook of Graph Grammars and Computing by Graph Transformation, ch. 4, vol. 1: Foundations, World Scientific, Singapore (1997)
4. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theoretical Computer Science 256(1–2), 63–92 (2001)
5. Löwe, M.: Algebraic approach to single-pushout graph transformation. Theoretical Computer Science 109, 181–224 (1993)
6. Rensink, A., Distefano, D.: Abstract graph transformation. In: Proc. of SVV 2005 (3rd International Workshop on Software Verification and Validation). ENTCS, vol. 157.1, pp. 39–59 (2005)
7. Robertson, N., Seymour, P.: Graph minors. XX. Wagner's conjecture. Journal of Combinatorial Theory, Series B 92(2), 325–357 (2004)
8. Robertson, N., Seymour, P.: Graph minors. XXIII. Nash-Williams' immersion conjecture (2006) (submitted for publication), `http://www.math.princeton.edu/~pds/papers/GM23/GM23.pdf`
9. Saksena, M., Wibling, O., Jonsson, B.: Graph grammar modeling and verification of ad hoc routing protocols. In: Proc. of TACAS 2008. LNCS, vol. 4963, pp. 18–32. Springer, Heidelberg (2008)