A General Framework for Well-Structured Graph Transformation Systems*

Barbara König and Jan Stückrath

Universität Duisburg-Essen, Germany
{barbara_koenig, jan.stueckrath}@uni-due.de

Abstract. Graph transformation systems (GTSs) can be seen as well-structured transition systems (WSTSs), thus obtaining decidability results for certain classes of GTSs. In earlier work it was shown that well-structuredness can be obtained using the minor ordering as a well-quasi-order. In this paper we extend this idea to obtain a general framework in which several types of GTSs can be seen as (restricted) WSTSs. We instantiate this framework with the subgraph ordering and the induced subgraph ordering and apply it to analyse a simple access rights management system.

1 Introduction

Well-structured transition systems [2,9] are one of the main sources for decidability results for infinite-state systems. They equip a state space with a quasi-order, which must be a well-quasi-order (wqo) and a simulation relation for the transition relation. If a system can be seen as a WSTS, one can decide the coverability problem, i.e., the problem of verifying whether, from a given initial state, one can reach a state that covers a final state, i.e., is larger than the final state with respect to the chosen order. Often, these given final states, and all larger states, are considered to be error states and one can hence check whether an error state is reachable. Large classes of infinite-state systems are well-structured, for instance (unbounded) Petri nets and certain lossy systems. For these classes of systems the theory provides a generic backwards reachability algorithm.

A natural specification language for concurrent, distributed systems with a variable topology are graph transformation systems [19] and they usually generate infinite state spaces. In those systems states are represented by graphs and state changes by (local) transformation rules, consisting of a left-hand and a right-hand side graph. In [11] it was shown how lossy GTSs with edge contraction rules can be viewed as WSTSs with the graph minor ordering [17,18] and the theory was applied to verify a leader election protocol and a termination detection protocol [4]. The technique works for arbitrary (hyper-)graphs, i.e. the state space is not restricted to certain types of graphs. On the other hand, in order to obtain well-structuredness, we can only allow certain rule sets, for instance one has to require an edge contraction rule for each edge label.

* Research partially supported by DFG project GaReV.

In order to make the framework more flexible we now consider other wqos, different from the minor ordering: the subgraph ordering and the induced subgraph ordering. The subgraph ordering and a corresponding WSTS were already studied in [3], but without the backwards search algorithm. Furthermore, we already mentioned the decidability result in the case of the subgraph ordering in [4], but did not treat it in detail and did not consider it as an instance of a general framework.

In contrast to the minor ordering, the subgraph ordering is not a wqo on the set of all graphs, but only on those graphs where the length of undirected paths is bounded [6]. This results in a trade-off: while the stricter order allows us to consider all possible sets of graph transformation rules in order to obtain a decision procedure, we have to make sure to consider a system where only graphs satisfying this restriction are reachable. Even if this condition is not satisfied, the procedure can yield useful partial coverability results. Also, it often terminates without excluding graphs not satisfying the restriction (this is also the case for our running example), producing exact results. We make these considerations precise by introducing *Q-restricted* WSTSs, where the order need only be a wqo on Q. In general, one wants Q to be as large as possible to obtain stronger statements.

It turns out that the results of [11] can be transferred to this new setting. Apart from the minor ordering and the subgraph ordering, there are various other wqos that could be used [8], leading to different classes of systems and different notions of coverability. In order to avoid redoing the proofs for every case, we here introduce a general framework which works for the case where the partial order can be represented by graph morphisms, which is applicable to several important cases. Especially, we state conditions required to perform the backwards search. We show that the case of the minor ordering can be seen as a special instance of this general framework and show that the subgraph and the induced subgraph orderings are also compatible. Finally we present an implementation and give runtime results. For the proofs we refer the reader to the extended version of this paper [15].

2 Preliminaries

2.1 Well-structured Transition Systems

We define an extension to the notion of WSTS as introduced in [2,9], a general framework for decidability results for infinite-state systems, based on well-quasi-orders.

Definition 1 (Well-quasi-order and upward closure). *A quasi-order $\leq$ (over a set X) is a* well-quasi-order (wqo) *if for any infinite sequence $x_0, x_1, x_2, \ldots$ of elements of X, there exist indices $i < j$ with $x_i \leq x_j$.*

An upward-closed set *is any set $I \subseteq X$ such that $x \leq y$ and $x \in I$ implies $y \in I$. For a subset $Y \subseteq X$, we define its upward closure $\uparrow Y = \{x \in X \mid \exists y \in Y : y \leq x\}$. Then, a* basis *of an upward-closed set I is a set I_B such that $I = \uparrow I_B$.*

A downward-closed set*, downward closure and a basis of a downward-closed set can be defined analogously.*

The definition of wqos gives rise to properties which are important for the correctness and termination of the backwards search algorithm presented later.

Lemma 1. *Let $\leq$ be a wqo, then the following two statements hold:*

1. *Any upward-closed set I has a finite basis.*
2. *For any infinite, increasing sequence of upward-closed sets $I_0 \subseteq I_1 \subseteq I_2 \subseteq \ldots$ there exists an index $k \in \mathbb{N}$ such that $I_i = I_{i+1}$ for all $i \geq k$.*

A Q-restricted WSTS is a transition system, equipped with a quasi-order, such that the quasi-order is a (weak) simulation relation on all states and a wqo on a restricted set of states Q.

Definition 2 (Q-restricted well-structured transition system). *Let S be a set of states and let Q be a downward closed subset of S, where membership is decidable. A* Q-restricted well-structured transition system (Q-restricted WSTS) *is a transition system $\mathcal{T} = (S, \Rightarrow, \leq)$, where the following conditions hold:*

Ordering: *$\leq$ is a quasi-order on S and a wqo on Q.*
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 $s_2 \leq t_2$.*

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

The presented Q-restricted WSTS are a generalization of WSTS and are identical to the classical definition, when $Q = S$. We will show how well-known results for WSTS can be transfered to Q-restricted WSTS. For Q-restricted WSTS there are two coverability problems of interest. The *(general) coverability problem* is to decide, given two states $s, t \in S$, whether there is a sequence of transitions $s \Rightarrow s_1 \Rightarrow \ldots \Rightarrow s_n$ such that $t \leq s_n$. The *restricted coverability problem* is to decide whether there is such a sequence for two $s, t \in Q$ with $s_i \in Q$ for $1 \leq i \leq n$. Both problems are undecidable in the general case (as a result of [4] and Proposition 5) but we will show that the well-known backward search for classical WSTS can be put to good use.

Given a set $I \subseteq S$ of states we denote by $\mathit{Pred}(I)$ the set of direct predecessors of I, i.e., $\mathit{Pred}(I) = \{s \in S \mid \exists s' \in I \colon s \Rightarrow s'\}$. Additionally, we use $\mathit{Pred}_Q(I)$ to denote the restriction $\mathit{Pred}_Q(I) = \mathit{Pred}(I) \cap Q$. Furthermore, we define $\mathit{Pred}^*(I)$ as the set of all predecessors (in S) which can reach some state of I with an arbitrary number of transitions. To obtain decidability results, the sets of predecessors must be computable, i.e. a so-called effective pred-basis must exist.

Definition 3 (Effective pred-basis). *A Q-restricted 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 $\uparrow \mathit{Pred}(\uparrow\{s\})$. It has an* effective Q-pred-basis *if there exists an algorithm accepting any state $q \in Q$ and returning $pb_Q(q)$, a finite basis of $\uparrow \mathit{Pred}_Q(\uparrow\{q\})$.*

Whenever there exists an effective pred-basis, there also exists an effective Q-pred-basis, since we can use the downward closure of Q to prove $pb_Q(q) = pb(q) \cap Q$.

Let $(S, \Rightarrow, \leq)$ be a Q-restricted WSTS with an effective pred-basis and let $I \subseteq S$ be an upward-closed set of states with finite basis I_B. To solve the general coverability problem we compute the sequence $I_0, I_1, I_2 \ldots$ where $I_0 = I_B$ and $I_{n+1} = I_n \cup pb(I_n)$. If the sequence $\uparrow I_0 \subseteq \uparrow I_1 \subseteq \uparrow I_2 \subseteq \ldots$ becomes stationary, i.e. there is an m with $\uparrow I_m = \uparrow I_{m+1}$, then $\uparrow I_m = \uparrow Pred^*(I)$ and a state of I is coverable from a state s if and only if there exists an $s' \in I_m$ with $s' \leq s$. If $\leq$ is a wqo on S, by Lemma 1 every upward-closed set is finitely representable and every sequence becomes stationary. However, in general the sequence might not become stationary if $Q \neq S$, in which case the problem becomes semi-decidable, since termination is no longer guaranteed (although correctness is).

The restricted coverability problem can be solved in a similar way, if an effective Q-pred-basis exists. Let $I^Q \subseteq S$ be an upward closed set of states with finite basis $I_B^Q \subseteq Q$. We compute the sequence $I_0^Q, I_1^Q, I_2^Q, \ldots$ with $I_0^Q = I_B^Q$ and $I_{n+1}^Q = I_n^Q \cup pb_Q(I_n^Q)$. Contrary to the general coverability problem, the sequence $\uparrow I_0^Q \cap Q \subseteq \uparrow I_1^Q \cap Q \subseteq \uparrow I_2^Q \cap Q \subseteq \ldots$ is guaranteed to become stationary according to Lemma 1. Let again m be the first index with $\uparrow I_m^Q = \uparrow I_{m+1}^Q$, and set $\Rightarrow_Q = (\Rightarrow \cap\, Q \times Q)$. We obtain the following result, of which the classical decidability result of [9] is a special case.

Theorem 1 (Coverability problems). *Let $T = (S, \Rightarrow, \leq)$ be a Q-restricted WSTS with a decidable order $\leq$.*

(i) If T has an effective pred-basis and $S = Q$, the general and restricted coverability problems coincide and both are decidable.
(ii) If T has an effective Q-pred-basis, the restricted coverability problem is decidable if Q is closed under reachability.
(iii) If T has an effective Q-pred-basis and I_m^Q is the limit as described above, then: if $s \in \uparrow I_m^Q$, then s covers a state of I^Q in $\Rightarrow$ (general coverability). If $s \notin \uparrow I_m^Q$, then s does not cover a state of I^Q in $\Rightarrow_Q$ (no restricted coverability).
(iv) If T has an effective pred-basis and the sequence I_n becomes stationary for $n = m$, then: a state s covers a state of I if and only if $s \in \uparrow I_m$.

Thus, if T is a Q-restricted WSTS and the "error states" can be represented as an upward-closed set I, then the reachability of an error state of I can be determined as described above, depending on which of the cases of Theorem 1 applies. Note that it is not always necessary to compute the limits I_m or I_m^Q, since $\uparrow I_i \subseteq \uparrow I_m$ (and $\uparrow I_i^Q \subseteq \uparrow I_m^Q$) for any $i \in \mathbb{N}_0$. Hence, if $s \in \uparrow I_i$ (or $s \in \uparrow I_i^Q$) for some i, then we already know that s covers a state of I (or of I^Q) in $\Rightarrow$.

2.2 Graph Transformation Systems

In the following we define the basics of hypergraphs and GTSs as a special form of transition systems where the states are hypergraphs and the rewriting rules

are hypergraph morphisms. We prefer hypergraphs over directed or undirected graphs since they are more convenient for system modelling.

Definition 4 (Hypergraph). *Let Λ be a finite sets of edge labels and $ar\colon \Lambda \to \mathbb{N}$ a function that assigns an arity to each label. A* (Λ-)hypergraph *is a tuple (V_G, E_G, c_G, l_G^E) 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 an (ordered) connection function and $l_G^E\colon E_G \to \Lambda$ is an edge labelling function. We require that $|c_G(e)| = ar(l_G^E(e))$ for each edge $e \in E_G$.*

An edge e is called incident *to a node v (and vice versa) if v occurs in $c_G(e)$.*

From now on we will often call hypergraphs simply graphs. An (elementary) *undirected path* of length n in a hypergraph is an alternating sequence $v_0, e_1, v_1, \ldots, v_{n-1}, e_n, v_n$ of nodes and edges such that for every index $1 \leq i \leq n$ both nodes v_{i-1} and v_i are incident to e_i and the undirected path contains all nodes and edges at most once. Note that there is no established notion of directed paths for hypergraphs, but our definition gives rise to undirected paths in the setting of directed graphs (which are a special form of hypergraphs).

Definition 5 (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 incident to it.* Total morphisms *are denoted by an arrow of the form $\to$.*

For simplicity we will drop the subscripts and write φ instead of φ_V and φ_E. We call two graphs G_1, G_2 isomorphic if there exists a total bijective morphism $\varphi : G_1 \to G_2$.

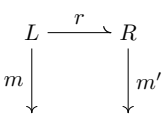
Graph rewriting relies on the notion of *pushouts*. It is known that pushouts of partial graph morphisms always exist and are unique up to isomorphism. Intuitively, for morphisms $\varphi : G_0 \rightharpoonup G_1$, $\psi : G_0 \rightharpoonup G_2$, the pushout is obtained by gluing the two graphs G_1, G_2 over the common interface G_0 and by deleting all elements which are undefined under φ or ψ (for a formal definition see [19]).

We will take pushouts mainly in the situation described in Definition 6 below, where r (the rule) is partial and connects the left-hand side L and the right-hand side R. It is applied to a graph G via a total match m. In order to ensure that the resulting morphism m' (the co-match of the right-hand side in the resulting graph) is also total, we have to require a match m to be *conflict-free* wrt. r, i.e., if there are two elements x, y of L with $m(x) = m(y)$ either $r(x), r(y)$ are both defined or both undefined. Here we consider a graph rewriting approach called the *single-pushout approach (SPO)* [7], since it relies on one pushout square, and restrict to conflict-free matches.

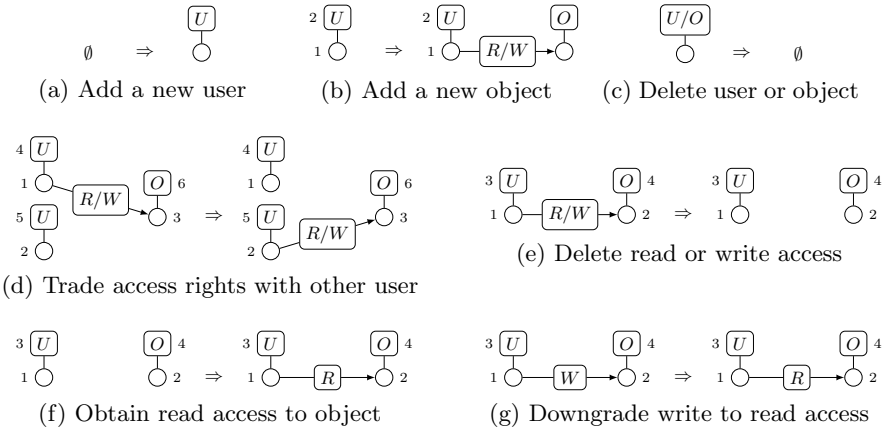
Definition 6 (Graph rewriting). *A* rewriting rule *is a partial morphism $r\colon L \rightharpoonup R$, where L is called left-hand and R right-hand side. A* match *(of r) is a total morphism $m\colon L \to G$, conflict-free wrt. r. Given a rule and a match, a*

rewriting step *or rule application is given by a pushout diagram as shown below, resulting in the graph H.*

A graph transformation system (GTS) *is a finite set of rules $\mathcal{R}$. Given a fixed set of graphs $\mathcal{G}$, a* graph transition system *on $\mathcal{G}$ generated by a graph transformation system $\mathcal{R}$ is represented by a tuple $(\mathcal{G}, \Rightarrow)$ where $\mathcal{G}$ is the set of states and $G \Rightarrow G'$ if and only if $G, G' \in \mathcal{G}$ and G can be rewritten to G' using a rule of $\mathcal{R}$.*



Later we will have to apply rules backwards, which means that it is necessary to compute so-called pushout complements, i.e., given r and m' above, we want to obtain G (such that m is total and conflict-free). How this computation can be performed in general is described in [10]. Note that pushout complements are not unique and possibly do not exist for arbitrary morphisms. For two partial morphisms the number of pushout complements may be infinite.



(a) Add a new user (b) Add a new object (c) Delete user or object

(d) Trade access rights with other user (e) Delete read or write access

(f) Obtain read access to object (g) Downgrade write to read access

Fig. 1: A GTS modelling a multi-user system

Example 1. To illustrate graph rewriting we model a multi-user system as a GTS (see Figure 1) inspired by [13]. A graph contains user nodes, indicated by unary U-edges, and object nodes, indicated by unary O-edges. Users can have read (R) or write (W) access rights regarding objects indicated by a (directed) edge. Note that binary edges are depicted by arrows, the numbers describe the rule morphisms and labels of the form R/W represent two rules, one with R-edges and one with W-edges.

The users and objects can be manipulated by rules for adding new users (Fig. 1a), adding new objects with read or write access associated with a user (Fig. 1b) and deleting users or objects (Fig. 1c). Both read and write access can be traded between users (Fig. 1d) or dropped (Fig. 1e). Additionally users can

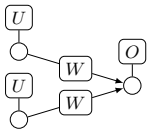


Fig. 2: An undesired state in the multi-user system

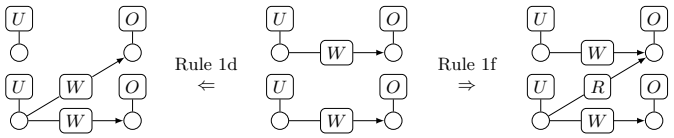


Fig. 3: Example of two rule applications

downgrade their write access to a read access (Fig. 1g) and obtain read access of arbitrary objects (Fig. 1f).

In a multi-user system there can be arbitrary many users with read access to an object, but at most one user may have write access. This means especially that any configuration of the system containing the graph depicted in Figure 2 is erroneous.

An application of the Rules 1d and 1f is shown in Figure 3. In general, nodes and edges on which the rule morphism r is undefined are deleted and nodes and edges of the right-hand side are added if they have no preimage under r. In the case of non-injective rule morphisms, nodes or edges with the same image are merged. Finally, node deletion results in the deletion of all incident edges (which would otherwise be left dangling). For instance, if Rule 1c is applied, all read/write access edges attached to the single deleted node will be deleted as well.

3 GTS as WSTS: A General Framework

In this section we state some sufficient conditions such that the coverability problems for $\mathcal{Q}$-restricted well-structured GTS can be solved in the sense of Theorem 1 (in the following we use $\mathcal{Q}$ to emphasize that $\mathcal{Q}$ is a set of graphs). We will also give an appropriate backward algorithm. The basic idea is to represent the wqo by a given class of morphisms.

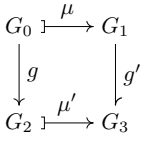
Definition 7 (Representable by morphisms). *Let $\sqsubseteq$ be a quasi-order that satisfies $G_1 \sqsubseteq G_2$, $G_2 \sqsubseteq G_1$ for two graphs G_1, G_2 if and only if G_1, G_2 are isomorphic, i.e., $\sqsubseteq$ is anti-symmetric up to isomorphism.*

We call $\sqsubseteq$ representable by morphisms *if there is a class of (partial) morphisms $\mathcal{M}_\sqsubseteq$ such that for two graphs G, G' it holds that $G' \sqsubseteq G$ if and only if there is a morphism $(\mu\colon G \rightarrowtail G') \in \mathcal{M}_\sqsubseteq$. Furthermore, for $(\mu_1 : G_1 \rightarrowtail G_2), (\mu_2 : G_2 \rightarrowtail G_3) \in \mathcal{M}_\sqsubseteq$ it holds that $\mu_2 \circ \mu_1 \in \mathcal{M}_\sqsubseteq$, i.e., $\mathcal{M}_\sqsubseteq$ is closed under composition. We call such morphisms μ* order morphisms*.*

The intuition behind an order morphism is the following: whenever there is an order morphism from G to G', we usually assume that G' is the smaller graph that can be obtained from G by some form of node deletion, edge deletion or edge contraction. For any graphs G (which represent all larger graphs) we can

now compose rules and order morphisms to simulate a co-match of a rule to some graph larger than G. However, for this construction to yield correct results, the order morphisms have to satisfy the following two properties.

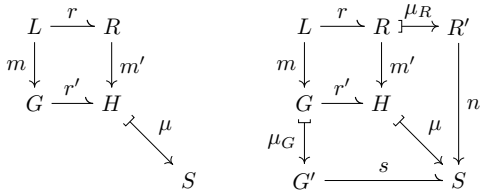
Definition 8 (Pushout preservation). *We say that a set of order morphisms $\mathcal{M}_{\sqsubseteq}$ is* preserved by total pushouts *if the following holds: if $(\mu\colon G_0 \rightarrowtail G_1) \in \mathcal{M}_{\sqsubseteq}$ is an order morphism and $g\colon G_0 \to G_2$ is total, then the morphism μ' in the pushout diagram on the right is an order morphism of $\mathcal{M}_{\sqsubseteq}$.*



The next property is needed to ensure that every graph G, which is rewritten to a graph H larger than S, is represented by a graph G' obtained by a backward rewriting step from S, i.e. the backward step need not be applied to H.

Definition 9 (Pushout closure). *Let $m\colon L \to G$ be total and conflict-free wrt. $r\colon L \rightharpoonup R$. A set of order morphisms is called* pushout closed *if the following holds: if the diagram below on the left is a pushout and $\mu\colon H \rightarrowtail S$ an order morphism, then there exist graphs R' and G' and order morphisms $\mu_R\colon R \rightarrowtail R'$, $\mu_G\colon G \rightarrowtail G'$, such that:*

1. *the diagram below on the right commutes and the outer square is a pushout.*
2. *the morphisms $\mu_G \circ m\colon L \to G'$ and $n\colon R' \to S$ are total and $\mu_G \circ m$ is conflict-free wrt. r.*



We now present a generic backward algorithm for (partially) solving both coverability problems. The procedure has two variants, which both require a GTS, an order and a set of final graphs to generate a set of minimal representatives of graphs covering a final graph. The first variant computes the sequence $I_n^{\mathcal{Q}}$ and restricts the set of graphs to ensure termination. It can be used for cases (i), (ii) and (iii) of Theorem 1, while the second variant computes I_n (without restriction) and can be used for cases (i) and (iv).

Procedure 1 (Computation of the ($\mathcal{Q}$-)pred-basis).
Input: A set $\mathcal{R}$ of graph transformation rules, a quasi-order $\sqsubseteq$ on all graphs which is a wqo on a downward-closed set $\mathcal{Q}$ and a finite set of final graphs $\mathcal{F}$, satisfying:

- The transition system generated by the rule set $\mathcal{R}$ is a $\mathcal{Q}$-restricted WSTS with respect to the order $\sqsubseteq$.
- The order $\sqsubseteq$ is representable by a class of morphisms $\mathcal{M}_{\sqsubseteq}$ (Definition 7) and this class satisfies Definitions 8 and 9.

– *Variant 1.* The set of minimal pushout complements restricted to $\mathcal{Q}$ with respect to $\sqsubseteq$ is computable, for all pairs of rules and co-matches (it is automatically finite).
Variant 2. The set of minimal pushout complements with respect to $\sqsubseteq$ is finite and computable, for all pairs of rules and co-matches.

Preparation: Generate a new rule set $\mathcal{R}'$ from $\mathcal{R}$ in the following way: for every rule $(r : L \rightharpoonup R) \in \mathcal{R}$ and every order morphism $\mu\colon R \rightarrowtail \overline{R}$ add the rule $\mu \circ r$ to $\mathcal{R}'$. (Note that it is sufficient to take a representative $\overline{R}$ for each of the finitely many isomorphism classes, resulting in a finite set $\mathcal{R}'$.) Start with the working set $\mathcal{W} = \mathcal{F}$ and apply the first backward step.
Backward Step: Perform backward steps until the sequence of working sets $\mathcal{W}$ becomes stationary. The following substeps are performed in one backward step for each rule $(r : L \rightharpoonup R) \in \mathcal{R}'$:

1. For a graph $G \in \mathcal{W}$ compute all total morphisms $m' : R \to G$ (co-matches of R in G).
2. *Variant 1.* For each such morphism m' calculate the set $\mathcal{G}_{poc}$ of minimal pushout complement objects of m' with rule r, which are also elements of $\mathcal{Q}$.
Variant 2. Same as Variant 1, but calculate *all* minimal pushout complements, without the restriction to $\mathcal{Q}$.
3. Add all remaining graphs in $\mathcal{G}_{poc}$ to $\mathcal{W}$ and minimize $\mathcal{W}$ by removing all graphs G' for which there is a graph $G'' \in \mathcal{W}$ with $G' \neq G''$ and $G'' \sqsubseteq G'$.

Result: The resulting set $\mathcal{W}$ contains minimal representatives of graphs from which a final state is coverable (cf. Theorem 1).

The reason for composing rule morphisms with order morphisms when doing the backwards step is the following: the graph G, for which we perform the step, might not contain a right-hand side R in its entirety. However, G can represent graphs that do contain R and hence we have to compute the effect of applying the rule backwards to all graphs represented by G. Instead of enumerating all these graphs (which are infinitely many), we simulate this effect by looking for matches of right-hand sides modulo order morphisms. We show that the procedure is correct by proving the following lemma.

Proposition 1. *Let $pb_1()$ and $pb_2()$ be a single backward step of Procedure 1 for Variant 1 and 2 respectively. For each graph S, $pb_1(S)$ is an effective $\mathcal{Q}$-pred-basis and $pb_2(S)$ is an effective pred-basis.*

4 Well-quasi Orders for Graph Transformation Systems

4.1 Minor Ordering

We first instantiate the general framework with the minor ordering, which was already considered in [11]. The minor ordering is a well-known order on graphs, which is defined as follows: a graph G is a minor of G' whenever G can be

obtained from G' by a series of node deletions, edge deletions and edge contractions, i.e. deleting an edge and merging its incident nodes according to an arbitrary partition. Robertson and Seymour showed in a seminal result that the minor ordering is a wqo on the set of all graphs [17], even for hypergraphs [18], thus case (i) of Theorem 1 applies. In [11,12] we showed that the conditions for WSTS are satisfied for a restricted set of GTS by introducing minor morphisms and proving a result analogous to Proposition 1, but only for this specific case. The resulting algorithm is a special case of both variants of Procedure 1.

Proposition 2 ([11]). *The coverability problem is decidable for every GTS if the minor ordering is used and the rule set contains edge contraction rules for each edge label.*

4.2 Subgraph Ordering

In this paper we will show that the subgraph ordering and the induced subgraph ordering satisfy the conditions of Procedure 1 for a restricted set of graphs and are therefore also compatible with our framework. For the subgraph ordering we already stated a related result (but for injective instead of conflict-free matches) in [4], but did not yet instantiate a general framework.

Definition 10 (Subgraph). *Let G_1, G_2 be graphs. G_1 is a subgraph of G_2 (written $G_1 \subseteq G_2$) if G_1 can be obtained from G_2 by a sequence of deletions of edges and isolated nodes. We call a partial morphism $\mu\colon G \rightarrowtail S$ a* subgraph morphism *if and only if it is injective on all elements on which it is defined and surjective.*

It can be shown that the subgraph ordering is representable by subgraph morphisms, which satisfy the necessary properties. Using a result from Ding [6] we can show that the set $\mathcal{G}_k$ of hypergraphs where the length of every undirected path is bounded by k, is well-quasi-ordered by the subgraph relation. A similar result was shown by Meyer for depth-bounded systems in [16]. Note that we bound undirected path lengths instead of directed path lengths. For the class of graphs with bounded directed paths there exists a sequence of graphs violating the wqo property (a sequence of circles of increasing length, where the edge directions alternate along the circle).

Since every GTS satisfies the compatibility condition of Definition 2 naturally, we obtain the following result.

Proposition 3 (WSTS wrt. the subgraph ordering). *Let k be a natural number. Every graph transformation system forms a $\mathcal{G}_k$-restricted WSTS with the subgraph ordering.*

The set of minimal pushout complements (not just restricted to $\mathcal{G}_k$) is always finite and can be computed as in the minor case.

Proposition 4. *Every $\mathcal{G}_k$-restricted well-structured GTS with the subgraph order has an effective pred-basis and the (decidability) results of Theorem 1 apply.*

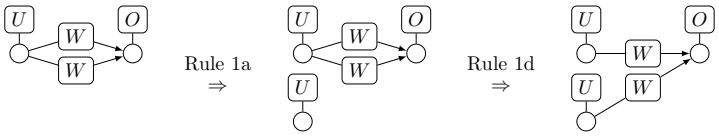
By a simple reduction from the reachability problem for two counter machines, we can show that the restricted coverability problem is undecidable in the general case. Although we cannot directly simulate the zero test, i.e. negative application conditions are not possible, we can make sure that the rules simulating the zero test are applied correctly if and only if the bound k was not exceeded.

Proposition 5. *Let $k > 2$ be a natural number. The restricted coverability problem for $\mathcal{G}_k$-restricted well-structured GTS with the subgraph ordering is undecidable.*

Example 2. Now assume that an error graph is given and that a graph exhibits an error if and only if it contains the error graph as a subgraph. Then we can use Proposition 4 to calculate all graphs which lead to some error configuration.

For instance, let a multi-user system as described in Example 1 be given. Normally we have to choose a bound on the undirected path length to guarantee termination, but in this example Variant 2 of Procedure 1 terminates and we can solve coverability on the unrestricted transition system (see Theorem 1(iv)). The graph in Figure 2 represents the error in the system and by applying Procedure 1 we obtain a set of four graphs (one of which is the error graph itself), fully characterizing all predecessor graphs. We can observe that the error can only be reached from graphs already containing two W-edges going to a single object node. Hence, the error is not produced by the given rule set if we start with the empty graph and thus the system is correct.

Interestingly the backward search finds the leftmost graph below due to the depicted sequence of rule applications, which leads directly to the error graph. Thus, the error can occur even if a single user has two write access rights to an object, because of access right trading.



The other two graphs computed are shown below and represent states with "broken" structure (a node cannot be a user *and* an object). The left graph for instance can be rewritten to a graph larger than the left graph above, by a non-injective match of the rule in Figure 1d mapping both nodes 2 and 3 to the right node.

4.3 Induced Subgraph Ordering

As for the subgraph ordering in Section 4.2 our backward algorithm can also be applied to the induced subgraph ordering, where a graph G is considered as an

induced subgraph of G' if every edge in G' connecting only nodes also present in G, is contained in G as well. Unfortunately, this ordering is not a wqo even when bounding the longest undirected path in a graph, such that we also have to bound the multiplicity of edges between two nodes. Note that this restriction is implicitly done in [6] since Ding uses simple graphs.

Furthermore, since we do not know whether the induced subgraph ordering can be extended to a wqo on (a class of) hypergraphs, we here use only *directed graphs*, where each edge is connected to a sequence of exactly two nodes. For many applications directed graphs are sufficient for modelling, also for our examples, since unary hyperedges can simply be represented by loops.

At first, this order seems unnecessary, since it is stricter than the subgraph ordering and is a wqo on a more restricted set of graphs. On the other hand, it allows us to specify error graphs more precisely, since a graph $G \in \mathcal{F}$ does not represent graphs with additional edges between nodes of G. Furthermore one could equip the rules with a limited form of negative application conditions, still retaining the compatibility condition of Definition 2.

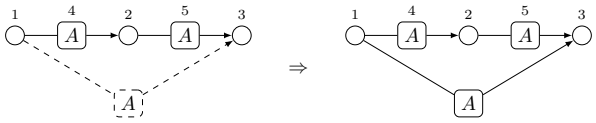
Definition 11 (Induced subgraph). *Let G_1, G_2 be graphs. G_1 is an induced subgraph of G_2 (written $G_1 \trianglelefteq G_2$) if G_1 can be obtained from G_2 by deleting a subset of the nodes and all incident edges. We call a partial morphism $\mu\colon G \rightarrowtriangle S$ an* induced subgraph morphism *if and only if it is injective for all elements on which is defined, surjective, and if it is undefined on an edge e, it is undefined on at least one node incident to e.*

Proposition 6 (WSTS wrt. the induced subgraph ordering). *Let n, k be natural numbers and let $\mathcal{G}_{n,k}$ be a set of directed, edge-labelled graphs, where the longest undirected path is bounded by n and every two nodes are connected by at most k parallel edges with the same label (bounded edge multiplicity). Every GTS forms a $\mathcal{G}_{n,k}$-restricted WSTS with the induced subgraph ordering.*

Proposition 7. *Every $\mathcal{G}_{n,k}$-restricted well-structured GTS with the induced subgraph order has an effective $\mathcal{G}_{n,k}$-pred-basis and the (decidability) results of Theorem 1 apply.*

The computation of minimal pushout complements in this case is considerably more complex, since extra edges have to be added, but we also obtain additional expressiveness. In general GTS with negative application conditions do not satisfy the compatibility condition with respect to the subgraph relation, but we show in the following example, that it may still be satisfied with respect to the induced subgraph relation.

Example 3. Let the following simple rule be given, where the negative application condition is indicated by the dashed edge, i.e. the rule is applicable if and only if there is a matching only for the solid part of the left-hand side and this matching cannot be extended to match also the dashed part.



Applied to a graph containing only A-edges, this rule calculates the transitive closure and will terminate at some point. This GTS satisfies the compatibility condition wrt. the induced subgraph ordering, since for instance a directed path of length two (the left-hand side) does not represent graphs where there is an edge from the first to the last node of the graph. Therefore we can use the induced subgraph ordering and our procedure to show that a graph containing two parallel A-edges can only be reached from graphs already containing two parallel A-edges.

The principle described in the example can be extended to all negative application conditions which forbid the existence of edges but not of nodes. This is the case, because if there is no edge between two nodes of a graph, there is also no edge between these two nodes in any larger graph. Hence if there is no mapping from the negative application condition into the smaller graph, there can also be none into the larger graph. Graphs violating the negative application condition are simply not represented by the smaller graph. Hence, all graph transformation rules with such negative application conditions satisfy the compatibility condition wrt. the induced subgraph ordering. The backward step has to be modified in this case by dropping all obtained graphs which do not satisfy one of the negative application conditions.

4.4 Implementation

We implemented Procedure 1 with support for the minor ordering as well as the subgraph ordering in the tool Uncover. The tool is written in C++ and designed in a modular way for easy extension with further orders. The sole optimization currently implemented is the omission of all rules that are also order morphisms. It can be shown that the backward application of such rules produces only graphs which are already represented.

Table 1 shows the runtime results of different case studies, namely a leader election protocol and a termination detection protocol (in an incorrect as well as a correct version), using the minor ordering, and the access rights management protocol described in Figure 1 as well as a public-private server protocol, using the subgraph order. It shows for each case the restricted graph set $\mathcal{Q}$, the variant of the procedure used (for the minor ordering they coincide), the runtime and the number of minimal graphs representing all predecessors of error graphs.

5 Conclusion

We have presented a general framework for viewing GTSs as restricted WSTSs. We showed that the work in [11] for the minor ordering can be seen as an instance

Table 1: Runtime result for different case studies

case study	wqo	graph set $\mathcal{Q}$	variant	time	#(error graphs)
Leader election	minor	all graphs	1 / 2	3s	38
Termination detection (faulty)	minor	all graphs	1 / 2	7s	69
Termination detection (correct)	minor	all graphs	1 / 2	2s	101
Rights management	subgraph	all graphs	2	1s	4
Public-private server ($l = 5$)	subgraph	path ≤ 5	1	1s	14
Public-private server ($l = 6$)	subgraph	path ≤ 6	1	16s	16

of this framework and we presented two additional instantiations, based on the subgraph ordering and the induced subgraph ordering. Furthermore we presented the management of read and write access rights as an example and discussed our implementation with very encouraging runtime results.

Currently we are working on an extension of the presented framework with rules, which can uniformly change the entire neighbourhood of nodes. In this case the computed set of predecessor graphs will be an over-approximation. More extensions are possible (possibly introducing over-approximations) and we especially plan to further investigate the integration of rules with negative application conditions as for the induced subgraph ordering. In [14] we introduced an extension with negative application conditions for the minor ordering, but still, the interplay of the well-quasi-order and conditions has to be better understood. Naturally, we plan to look for additional orders, for instance the induced minor and topological minor orderings [8] in order to see whether they can be integrated into this framework and to study application scenarios.

Related work. Related to our work is [3], where the authors use the subgraph ordering and a forward search to prove fair termination for depth-bounded systems. In [1] another wqo for well-structuring graph rewriting is considered, however only for graphs where every node has out-degree 1. It would be interesting to see whether this wqo can be integrated into our general framework. The work in [5] uses the induced subgraph ordering to verify broadcast protocols. There the rules are different from our setting: a left-hand side consists of a node and its entire neighbourhood of arbitrary size. Finally [20] uses a backwards search on graph patterns in order to verify an ad-hoc routing protocol, but not in the setting of WSTSs.

Acknowledgements: We would like to thank Roland Meyer, for giving us the idea to consider the subgraph ordering on graphs, and Giorgio Delzanno for several interesting discussions on wqos and WSTSs.

References

1. P. Aziz Abdulla, A. Bouajjani, J. Cederberg, F. Haziza, and A. Rezine. Monotonic abstraction for programs with dynamic memory heaps. In *Proc. of CAV '08*, pages 341–354. Springer, 2008. LNCS 5123.

2. P. Aziz Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay. General decidability theorems for infinite-state systems. In *Proc. of LICS '96*, pages 313–321. IEEE, 1996.
3. K. Bansal, E. Koskinen, T. Wies, and D. Zufferey. Structural counter abstraction. In *Proc. of TACAS '13*, TACAS'13, pages 62–77. Springer, 2013.
4. N. Bertrand, G. Delzanno, B. König, A. Sangnier, and J. Stückrath. On the decidability status of reachability and coverability in graph transformation systems. In *Proc. of RTA '12*, volume 15 of *LIPIcs*, pages 101–116. Schloss Dagstuhl – Leibniz Center for Informatics, 2012.
5. G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *Proc. CONCUR '10*, pages 313–327. Springer, 2010. LNCS 6269.
6. G. Ding. Subgraphs and well-quasi-ordering. *Jornal of Graph Theory*, 16:489–502, November 1992.
7. H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic approaches to graph transformation—part II: Single pushout approach and comparison with double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 4. World Scientific, 1997.
8. M.R. Fellows, D. Hermelin, and F.A. Rosamond. Well-quasi-orders in subclasses of bounded treewidth graphs. In *Proc. of IWPEC '09 (Parameterized and Exact Computation)*, pages 149–160. Springer, 2009. LNCS 5917.
9. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, April 2001.
10. M. Heumüller, S. Joshi, B. König, and J. Stückrath. Construction of pushout complements in the category of hypergraphs. In *Proc. of GCM '10 (Workshop on Graph Computation Models)*, 2010.
11. S. Joshi and B. König. Applying the graph minor theorem to the verification of graph transformation systems. In *Proc. of CAV '08*, pages 214–226. Springer, 2008. LNCS 5123.
12. S. Joshi and B. König. Applying the graph minor theorem to the verification of graph transformation systems. Technical Report 2012-01, Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, 2012.
13. M. Koch, L.V. Mancini, and F. Parisi-Presicce. Decidability of safety in graph-based models for access control. In *Proceedings of the 7th European Symposium on Research in Computer Security*, pages 229–243. Springer, 2002. LNCS 2502.
14. B. König and J. Stückrath. Well-structured graph transformation systems with negative application conditions. In *Proc. of ICGT '12*, pages 89–95. Springer, 2012. LNCS 7562.
15. B. König and J. Stückrath. A general framework for well-structured graph transformation systems, 2014. arXiv:1406.4782.
16. R. Meyer. *Structural Stationarity in the π-Calculus*. PhD thesis, Carl-von-Ossietzky-Universität Oldenburg, 2009.
17. N. Robertson and P. Seymour. Graph minors. XX. Wagner's conjecture. *Journal of Combinatorial Theory, Series B*, 92(2):325–357, 2004.
18. N. Robertson and P. Seymour. Graph minors XXIII. Nash-Williams' immersion conjecture. *Journal of Combinatorial Theory Series B*, 100:181–205, March 2010.
19. G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*. World Scientific, 1997.
20. M. Saksena, O. Wibling, and B. Jonsson. Graph grammar modeling and verification of ad hoc routing protocols. In *Proc. of TACAS '08*, pages 18–32. Springer, 2008. LNCS 4963.