Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Well-structured graph transformation systems ☆

Barbara König *, Jan Stückrath

Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, Lotharstraße 65, 47057 Duisburg, Germany

ARTICLE INFO

Article history:
Received 11 December 2014
Available online xxxx

ABSTRACT

Graph transformation systems (GTSs) can be seen as well-structured transition systems (WSTSs) and via well-structuredness it is possible to obtain decidability results for certain classes of GTSs. We present a generic framework, parameterized over the well-quasi-order (wqo), in which several types of GTSs can be seen as (restricted) WSTSs. We instantiate this framework with three orders: the minor ordering, the subgraph ordering and the induced subgraph ordering. Furthermore we consider two case studies where we apply the theory to analyze a leader election protocol and a simple access rights management system with our tool UNCOVER.

1. Introduction

Well-structured transition systems [3,4] 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. The latter condition is also known as compatibility condition or monotonicity requirement. If a system can be seen as a WSTS and some additional conditions are satisfied, 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 of [3,4] provides a generic backwards reachability algorithm.

A natural specification language for concurrent, distributed systems are graph transformation systems [5] and they usually generate infinite state spaces (even if one factors the state space through graph isomorphism). In those systems states are represented by (hyper-)graphs and state changes by (local) transformation rules, consisting of a left-hand and a right-hand side graph. Graph transformation systems are especially suitable for modeling systems with a variable topology and dynamic creation and deletion of objects.

When working with graphs, we have several orders at our disposal: for instance, the minor ordering, the subgraph ordering and the induced subgraph ordering, leading to different notions of coverability.

The minor ordering is a well-quasi-order for all graphs [6,7], however in order to obtain well-structuredness, we can only allow certain rule sets, for instance one can consider lossy graph transformation systems, where we require an edge contraction rule for each edge label.

☆ This paper is based on our CONCUR '14 paper [1] and also integrates some results which were first published in CAV '08 [2]. The research is partially supported by the DFG project GaReV (KO 2185/6).

* Corresponding author.
E-mail addresses: barbara_koenig@uni-due.de (B. König), jan.stueckrath@uni-due.de (J. Stückrath).

http://dx.doi.org/10.1016/j.ic.2016.03.005
0890-5401/

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Concerning the subgraph and the induced subgraph ordering, the compatibility condition of WSTSs holds for all rules and it is unnecessary to restrict to lossy systems. On the other hand, the subgraph and the induced subgraph ordering are well-quasi-orders not on the set of all graphs, but only on those graphs where the length of undirected paths is bounded [8]. (For the induced subgraph ordering we additionally have to bound edge multiplicity.) So, in order to obtain a decision procedure, we have 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 even if not all reachable graphs satisfy the restriction (as in our running example), still producing exact results. We make these considerations precise by introducing Q-*restricted* WSTSs, where the order need only be a wqo on Q, a subset of the state space. In general, one wants Q to be as large as possible to obtain stronger statements.

It is clear that there is no order that is superior over all the others, instead there is a trade-off: while the stricter order allows us to consider all graph transformation rules, we have to restrict the state space to a subset Q. Instead, a coarser order requires certain conditions on rule sets, but allows larger state spaces, possibly the set of all graphs. In order to avoid redoing the proofs for every case, we introduce here a flexible general framework which works for orders that can be represented by a class of graph morphisms. This is the case for the three orders mentioned above. Especially, we state the conditions required to perform the backwards search.

In the paper we present two case studies: the verification of a leader election protocol (via the minor ordering) and the verification of a simple access rights management system (via the subgraph ordering). The algorithms for the case of the subgraph and the minor ordering have been implemented in the tool Uncover.[1] We discuss the tool and give runtime results for our running examples as well as for other case studies.

2. Preliminaries

2.1. Well-structured transition systems

We define an extension to the notion of WSTSs as introduced in [3,4], a general framework for decidability results for infinite-state systems, based on well-quasi-orders. Our terminology concerning WSTSs is mainly based on [4].

Definition 1 *(Well-quasi-order and upward closure).* Let X be some set. A quasi-order $\leq$ over 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. *(See [9].) 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 $i \geq k$.*

The first property must hold, since an infinite minimal basis implies that the elements of this basis would contain an infinite sequence of elements violating the wqo property. If the second property would not hold, we could form such an infinite sequence by taking elements of $I_{i+1} \setminus I_i$ for each i.

A Q-restricted well-structured transition system (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 WSTSs).* Let S be a set of states and let Q be a downward closed subset of S, where membership is decidable, i.e. for every $s \in S$ we can decide whether $s \in Q$ or not. 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 transitions $s_1 \Rightarrow s_2$, there exists a sequence $t_1 \Rightarrow^* t_2$ of transitions such that $s_2 \leq t_2$.

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

[1] The tool is available via www.ti.inf.uni-due.de/research/tools/uncover/.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

The presented Q-restricted WSTSs are a generalization of WSTSs and are identical to the classical definition, whenever $Q = S$. We will show how well-known results for WSTSs can be transferred to Q-restricted WSTSs. For Q-restricted WSTSs 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 [10] and Proposition 10) but we will show that the well-known backward search for classical WSTSs can be put to good use.

Given a set $I \subseteq S$ of states we denote by $Pred(I)$ the set of direct predecessors of I, i.e., $Pred(I) = \{s \in S \mid \exists s' \in I : s \Rightarrow s'\}$. Additionally, we use $Pred_Q(I)$ to denote the restriction $Pred_Q(I) = Pred(I) \cap Q$. Furthermore, we define $Pred^*(I)$ as the set of all predecessors (in S) which can reach some state of I with an arbitrary number of transitions. Note that in general $Pred^*(I) \cap Q \neq Pred_Q^*(I)$.

To obtain decidability results, the predecessor sets 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 $\uparrow Pred(\uparrow\{s\})$. 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 $\uparrow Pred_Q(\uparrow\{q\})$.

If 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 [4] is a special case.

Theorem 1 *(Coverability problems). Let $T = (S, \Rightarrow, \leq)$ be a Q-restricted WSTS with a decidable order $\leq$, i.e. for two $s_1, s_2 \in S$ we can decide whether $s_1 \leq s_2$ or not.*

(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 via $\Rightarrow$ (general coverability). If $s \notin \uparrow I_m^Q$, then s does not cover a state of I^Q via $\Rightarrow_Q$ (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 $s \in \uparrow I_m$.*

Proof. (i) is just a reformulation of the decidability results for WSTSs. Similar for (ii), if Q is closed under reachability, a Q-restricted WSTSs can be seen as a WSTSs with state space Q.

We now consider (iii) where Q is not required to be closed under reachability. Assume that $s \in \uparrow I_m^Q$, where I_m^Q has been obtained by fixed-point iteration, starting with the upward-closed set I_B^Q and computing the sequence I_n^Q. By induction we show the existence of a sequence of transitions leading from s to some state in $\uparrow I_B^Q$. Obviously there is an $q_m \in I_m^Q$ with $q_m \leq s$ and by definition either $q_m \in I_{m-1}^Q$ or there are $q_{m-1} \in I_{m-1}^Q$ and q'_{m-1} with $q_m \Rightarrow q'_{m-1}$ and $q_{m-1} \leq q'_{m-1}$. In the latter case, because of the compatibility condition of Definition 2, there is a q''_{m-1} with $s \Rightarrow^* q''_{m-1}$ and $q_{m-1} \leq q'_{m-1} \leq q''_{m-1}$, i.e. s can reach an element of $\uparrow I_{m-1}^Q$. Since this argument holds for q''_{m-1} as well, the state s can ultimately reach a state $q''_0 \in \uparrow I_B^Q$. Note that it is possible that $q''_0 \in Q$, but it is not guaranteed that $q''_n \in Q$ for every n.

For the other statement assume that $s \notin \uparrow I_m^Q$ and assume that there exists a path $s = q_0 \Rightarrow_Q q_1 \Rightarrow_Q \cdots \Rightarrow_Q q_k \in \uparrow I_B^Q$. Note that the second assumption is trivially false if $s \notin Q$. We can show by induction and by definition of $pb_Q()$ that $q_i \in \uparrow I_{k-i}^Q$ and hence $q_0 \in \uparrow I_k^Q \subseteq \uparrow I_m^Q$, which leads to a contradiction.

The proof of case (iv) is straightforward by observing that the set I_m is an exact representation of all predecessors of I. □

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

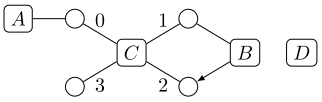


Fig. 1. Example of a hypergraph with four nodes and four edges.

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. For instance in a Petri modeling a mutual exclusion protocol the error states would be all markings with more than one token in the critical section. In this case the marking with two tokens in the critical section and none elsewhere represents all erroneous configurations, i.e. they are its upward closure. Thus, we only need to check coverability for this marking to check if the modeled protocol is erroneous.

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) via $\Rightarrow$.

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 labeled hypergraphs and the rewriting rules are hypergraph morphisms. We prefer hypergraphs over directed or undirected graphs since they are more convenient for system modeling. Note that directed graphs are a special case of hypergraphs.

In order to introduce graph rewriting, we need some category theory, in particular pushouts, which we will define below. For more details see [11].

Definition 4 *(Hypergraph).* Let Λ be a finite sets of edge labels and $ar\colon \Lambda \to \mathbb{N}_0$ a function that assigns an arity to each label. A *(Λ-)hypergraph* G 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 labeling 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 $c_G(e)$.

An example of a hypergraph can be seen in Fig. 1 consisting of four nodes (circles) and four edges (boxes) of arity 0, 1, 2 and 4. Edges with an arity of at least three (such as C) have numbered connections to nodes indicating their position in the sequence. Binary edges are drawn as directed edges where the source is the first and target the second element of the incident sequence of nodes. Edges with an even lesser arity are drawn as shown in Fig. 1.

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.

Definition 5 *(Partial graph morphism).* Let G, G' be (Λ-)hypergraphs. A *partial hypergraph morphism* (or simply *morphism*) $f\colon G \rightharpoonup G'$ consists of a pair of partial functions $(f_V : V_G \rightharpoonup V_{G'}, f_E : E_G \rightharpoonup E_{G'})$ such that for every $e \in E_G$ it holds that $l_G(e) = l_{G'}(f_E(e))$ and $f_V(c_G(e)) = c_{G'}(f_E(e))$ whenever $f_E(e)$ is defined (we use the extension of f_V to sequences in the straightforward way, i.e. component wise). Furthermore if a morphism is defined on an edge, it must be defined on all incident nodes. *Total morphisms*, where f_V and f_E are total, are denoted by an arrow of the form $\to$.

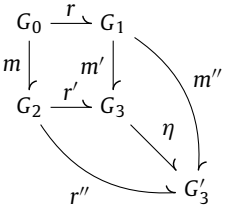
We call a partial morphism f surjective, if every node or edge of G' has a preimage in G. It is injective if there are no two nodes or edges $x_1 \neq x_2$, such that $f(x_1) = f(x_2)$ and $f(x_1), f(x_2)$ are both defined.

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

We transform graphs via the *single pushout approach (SPO)* [12] where a transformation consists of computing the pushout of graph morphisms. We will therefore first introduce the notion of pushouts and how theses can be constructed.

Definition 6 *(Pushout).* Let G_0, G_1, G_2 be graphs with partial graph morphisms $r\colon G_0 \rightharpoonup G_1$ and $m\colon G_0 \rightharpoonup G_2$. The graph G_3 together with $m'\colon G_1 \rightharpoonup G_3$ and $r'\colon G_2 \rightharpoonup G_3$ is a pushout of r, m if the following conditions are satisfied:

1. $m' \circ r = r' \circ m$.
2. For all $m''\colon G_1 \rightharpoonup G_3'$, $r''\colon G_2 \rightharpoonup G_3'$ satisfying $m'' \circ r = r'' \circ m$ there exists a unique morphism $\eta\colon G_3 \rightharpoonup G_3'$ such that $\eta \circ m' = m''$ and $\eta \circ r' = r''$.



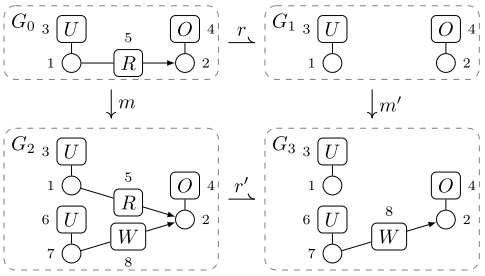


Fig. 2. Example of a pushout, i.e. an application of rule r to graph G_2.

It is known that pushouts of partial graph morphisms always exist and are unique up to isomorphism. A well-known construction exists, where pushouts are obtained by taking the disjoint union of G_1 and G_2 and factoring through an equivalence obtained from the morphisms r, m. Intuitively, G_1 and G_2 are glued over the common interface G_0, at the same time deleting all elements which are undefined under r or m. For the proof of the proposition see for instance [13].

Proposition 1 *(Pushout via equivalence classes). Let the (partial) morphisms $r\colon G_0 \rightharpoonup G_1$ and $m\colon G_0 \rightharpoonup G_2$ be given. 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 $r(x) \equiv m(x)$ 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 $r(x)$ or $m(x)$ are undefined. Similarly a class of edges is* valid *if the analogous condition holds and furthermore all nodes incident to these edges are contained in valid equivalence classes.*

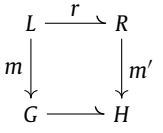
Then the pushout G_3 of r and m consists of all valid equivalence classes $[x]_\equiv$ as nodes and edges, where $l_{G_3}([e]_\equiv) = l_{G_i}(e)$ and $c_{G_3}([e]_\equiv) = [v_1]_\equiv \dots [v_k]_\equiv$ if $e \in E_{G_i}$ and $c_{G_i}(e) = v_1 \dots v_k$. The morphisms $m'\colon G_1 \rightharpoonup G_3$, $r'\colon G_2 \rightharpoonup G_3$ are defined as $m'(x) = [x]_\equiv$ and $r'(x) = [x]_\equiv$ for every valid $[x]_\equiv$, and undefined otherwise.

An example of a pushout can be seen in Fig. 2. The morphisms are indicated by numbering, i.e. an element of G_3 numbered i is the equivalence class containing all elements of G_1, G_2 also numbered i. The equivalence class of the edge numbered 5 is not valid and therefore deleted.

We will take pushouts mainly in the situation described in Definition 7, 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 the SPO (single-pushout) approach, since it relies on one pushout square, and restrict to conflict-free matches. SPO rewriting is suitable for our purposes for two reasons: first, it works with partial graph morphisms, which we will later also use to represent orders compatible with our framework. Second, SPO rewriting does not know inhibiting conditions, different from DPO (double-pushout) rewriting: if a node is deleted, all incident edges are automatically deleted as well. Edges that are implicitly deleted in this way, i.e., which are not deleted explicitly by the rule, are called *dangling edges*.

Definition 7 *(Graph transformation).* A *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 *transformation 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 $G \Rightarrow H$ if and only if $G, H \in \mathcal{G}$ and G can be transformed to H 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 [14], a revised version of the original GCM publication. 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, but we only need to compute those minimal wrt. the order used. This set is guaranteed to be finite if the order is a well-quasi-order.

Example 1. To illustrate graph rewriting we model a multi-user system as a GTS (see Fig. 3) inspired by [15]. 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,

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

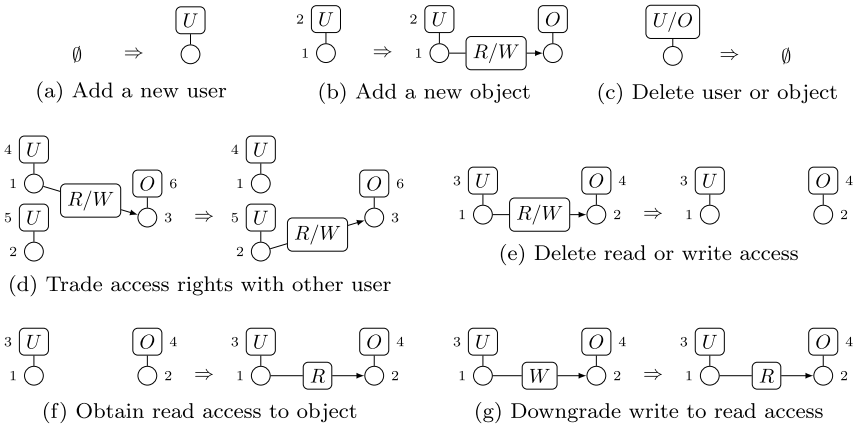


Fig. 3. A GTS modeling a multi-user system.

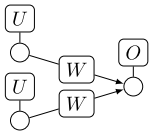


Fig. 4. An undesired state in the multi-user system.

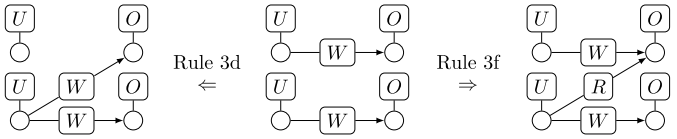


Fig. 5. Example of two rule applications.

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. 3a), adding new objects with read or write access associated with a user (Fig. 3b) and deleting users or objects (Fig. 3c). Both read and write access can be traded between users (Fig. 3d) or dropped (Fig. 3e). Additionally users can downgrade their write access to a read access (Fig. 3g) and obtain read access of arbitrary objects (Fig. 3f).

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 Fig. 4 is considered erroneous.

An application of the Rules 3d and 3f is shown in Fig. 5. 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 3c is applied, all read/write access edges attached to the single deleted node will be deleted as well.

3. GTSs as WSTSs: a general framework

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

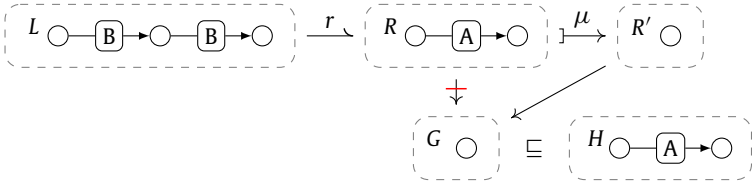
Definition 8 *(Representable by morphisms).* Let $\sqsubseteq$ be a quasi-order that satisfies $G_1 \sqsubseteq G_2$, $G_2 \sqsubseteq G_1$ for two graphs G_1, G_2 if and only if G_1, G_2 are isomorphic, i.e., $\sqsubseteq$ is anti-symmetric up to isomorphism.

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 $\mu\colon G \mapsto G'$ in $\mathcal{M}_\sqsubseteq$. Furthermore, we require that for $(\mu_1 : G_1 \mapsto G_2), (\mu_2 : G_2 \mapsto G_3) \in \mathcal{M}_\sqsubseteq$ it holds that $(\mu_2 \circ \mu_1) \in \mathcal{M}_\sqsubseteq$, i.e., $\mathcal{M}_\sqsubseteq$ is closed under composition, and for every G there are only finitely many $(\mu : G \mapsto G') \in \mathcal{M}_\sqsubseteq$ up to isomorphism. We call such morphisms μ *order morphisms*.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

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. Since this normally means that order morphisms are surjective, there are only finitely many morphisms up to isomorphism for a fixed domain graph G.

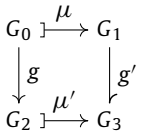
Given a graph G, in order to compute its pred-base, we have to generate all graphs that can be transformed to a graph larger than G, i.e. we need to apply rules backward, not just to G, but any larger graph. For this we compose rules and order morphisms to simulate the backward application to any graph in the upward closure of G, as shown below.



Although there is no co-match from R to G, there is a graph H larger than G to which a co-match of the rule exists. Since G represents its upward closure, we have to apply the rule backward to H as well. The enumeration of every such H is impossible, since there are infinitely many graphs. By forming $\mu \circ r$ we obtain a derived version of r which has a co-match to G and simulates a co-match to H. Note that the backwards application of $\mu \circ r$ will add all elements deleted by μ, since these parts are necessary to apply the rule (forward). With these "prepared" rules we only have to compute total co-matches to G to simulate the backwards application of the original rule r to the entire upward closure. This corresponds to the usual strategy for WSTSs: in a rule, one replaces the right-hand side by all objects which are smaller with respect to the given order.

To ensure the correctness of this approach, we need two properties: pushout preservation to ensure the soundness and pushout closure to ensure the completeness.

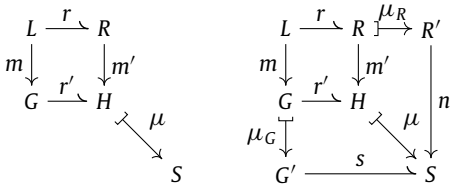
Definition 9 *(Pushout preservation).* We say that a class of order morphisms $\mathcal{M}_{\sqsubseteq}$ is *preserved by pushouts along total morphisms* if the following holds: if $\mu\colon G_0 \rightarrowtail G_1$ is an order morphism in $\mathcal{M}_{\sqsubseteq}$ and $g\colon G_0 \to G_2$ is total, then the morphism μ' in the pushout diagram on the right is an order morphism in $\mathcal{M}_{\sqsubseteq}$.



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 directly. In a sense, it can be seen as a compositionality property: if H can be obtained by composing G, R, then every minor of H can be obtained by composing minors of G, R.

Definition 10 *(Pushout closure).* A class of order morphisms is called *pushout closed* if the following holds: let $m\colon L \to G$ be total and conflict-free wrt. $r\colon L \rightharpoonup R$. 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 as input a GTS $\mathcal{R}$, an order $\sqsubseteq$ and a set of final graphs $\mathcal{F}$. Both variants generate as output a set of minimal representatives of graphs covering a final graph. The first variant computes the sequence $I_m^{\mathcal{Q}}$ 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_m (without restriction) and can be used for cases (i) and (iv). To ensure correctness, the input parameters need to satisfy the following conditions:

- The transition system generated by the rule set $\mathcal{R}$ needs to be a $\mathcal{Q}$-restricted WSTS with respect to the order $\sqsubseteq$.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

- The order $\sqsubseteq$ must be representable by a class of morphisms $\mathcal{M}_{\sqsubseteq}$ (Definition 8) and this class must satisfy the Definitions 9 and 10.

Furthermore, we also need some additional assumptions for the procedure to be effective. For instance, since the number of pushout complements is not necessarily finite, we need to restrict to a subset of pushout complements that are minimal wrt. the order used.

Definition 11 *(Minimal pushout complements).* Let $r\colon L \rightharpoonup R$ be a rule, $\mu\colon R \rightarrowtail R'$ an order morphism of an order $\sqsubseteq$ (representable by morphisms) and $m'\colon R' \to G$ a total co-match, as shown below. The *set of minimal pushout complements* of $\mu \circ r$ and m' is a set of graphs $\mathcal{G}_{poc}$, such that:

- every $G' \in \mathcal{G}_{poc}$ is a pushout complement of $\mu \circ r$, m' where the match $m\colon L \to G'$ is conflict-free wrt. r,
- for every pushout complement G'' satisfying the previous condition there is a $G' \in \mathcal{G}_{poc}$ with $G' \sqsubseteq G''$, and
- if there are $G', G'' \in \mathcal{G}_{poc}$ with $G' \sqsubseteq G''$, then $G' = G''$.

$$\begin{array}{ccc} L & \xrightarrow{r} & R \overset{\mu}{\rightarrowtail} R' \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle m'} \\ G' & \dashrightarrow & G \end{array}$$

The following algorithm for the computation of pred-bases has two variants: for restricted coverability (*variant 1*) and general coverability (*variant 2*). For *variant 1* the set of minimal pushout complements restricted to $\mathcal{Q}$ (i.e. only considering pushout complements in $\mathcal{Q}$) with respect to $\sqsubseteq$ needs to be computable for all pairs of rules and co-matches. This set is guaranteed to be finite if $\sqsubseteq$ is a wqo on $\mathcal{Q}$. For *variant 2* the set of minimal pushout complements with respect to $\sqsubseteq$ needs to be finite and computable, for all pairs of rules and co-matches. Additionally we need a procedure deciding $G_1 \sqsubseteq G_2$ for any two graphs G_1, G_2, and a procedure that can construct all order morphisms $G_1 \rightarrowtail G_2$, up to isomorphism, for a given graph G_1. These procedures are enough to implicitly represent the order used.

Procedure 1 *(Computation of the ($\mathcal{Q}$-)pred-basis).*
Input: A finite 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 conditions mentioned above.
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 $(\mu \circ r : L \rightharpoonup R) \in \mathcal{R}'$:

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 complements of m' with rule $\mu \circ r$ (where the match is conflict-free wrt. r), which are also elements of $\mathcal{Q}$.
   Variant 2. Same as Variant 1, but calculate *all* minimal pushout complements (where the match is conflict-free wrt. r), without restricting to $\mathcal{Q}$.
3. Add all 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: After becoming stationary, the resulting set $\mathcal{W}$ contains minimal representatives of graphs from which a final state is coverable (cf. Theorem 1).

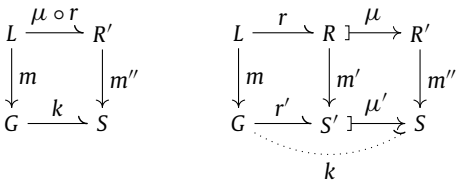
For order morphisms satisfying the Definitions 9 and 10 we show that the procedure is correct by proving the following two lemmas.

Lemma 2. *The sets generated by $pb_1(S)$ and $pb_2(S)$ are both finite subsets of $Pred(\uparrow\{S\})$ and $pb_1(S) \subseteq \mathcal{Q}$.*

Proof. By the conditions of Procedure 1, the sets of minimal pushout complements – in the case of $pb_1(S)$ restricted to $\mathcal{Q}$ – are finite and computable. Since the set of rules is also finite, $pb_1(S)$ and $pb_2(S)$ are finite as well. Every non-minimal pushout complement in $\mathcal{Q}$ is represented by a minimal one in $\mathcal{Q}$, because of the downward closure of $\mathcal{Q}$. Thus, $pb_1(S) \subseteq \mathcal{Q}$ holds.

Let $G \in pb_1(S) \cup pb_2(S)$ be a graph generated by one of the procedures. Then there is a rule $r\colon L \to R$, an order morphism $\mu\colon R \rightarrowtail R'$ and a conflict-free match $m\colon L \to G$, such that the left diagram below is a pushout.

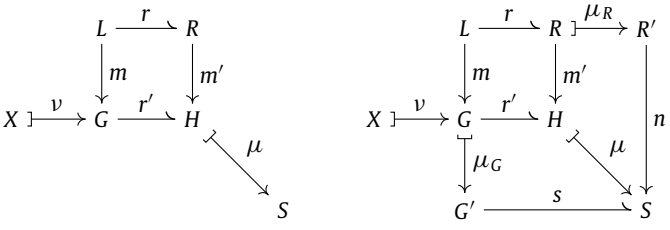
Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005



Let $m' \colon R \to S'$, $r' \colon G \rightharpoonup S'$ be the pushout of m, r. Because the outer diagram on the right commutes, there is a unique morphism $\mu' \colon S' \rightarrowtail S$. The left and the outer square are both pushouts and therefore also the right square is a pushout. Since m is total and conflict-free, m' is also total. By assumption $\mathcal{M}_{\sqsubseteq}$ is preserved by pushouts along total morphisms, thus μ' is in fact an order morphism. This means that G can be rewritten to some graph S' larger than S, hence $G \in \mathit{Pred}(\uparrow S)$. □

Lemma 3. *It holds that* $\uparrow pb_1(S) \supseteq \uparrow \mathit{Pred}_{\mathcal{Q}}(\uparrow\{S\})$ *and* $\uparrow pb_2(S) \supseteq \uparrow \mathit{Pred}(\uparrow\{S\})$.

Proof. Let X be an element of $\uparrow \mathit{Pred}(\uparrow\{S\})$. Then there is a minimal representative $G \in \mathit{Pred}(\uparrow\{S\})$ with $G \sqsubseteq X$ and a rule $r \colon L \rightharpoonup R$ rewriting G with a conflict-free match m to some element H of $\uparrow\{S\}$. According to Definition 10 the left diagram below can be extended to the right diagram below.



Since the outer square is a pushout, G' is a pushout complement object. Thus, a graph $G'' \sqsubseteq G'$ will be obtained by the procedure $pb_2()$ in Step 2 using the rule $\mu_R \circ r$. Summarized, this means that $pb_2()$ computes a graph G'' for every graph X such that $G'' \sqsubseteq G' \sqsubseteq G \sqsubseteq X$, i.e. every X is represented by an element of $pb_2(S)$.

Now assume $X \in \uparrow \mathit{Pred}_{\mathcal{Q}}(\uparrow\{S\})$. By definition, the minimal representative G is an element of $\mathcal{Q}$. We obtain $G' \in \mathcal{Q}$, due to the downward closure of $\mathcal{Q}$. Thus, the procedure $pb_1()$ will compute a graph $G'' \sqsubseteq G'$ (with $G'' \in \mathcal{Q}$), i.e. every X is represented by an element of $pb_1(S)$. □

Proposition 2. *Let* $pb_1()$ *and* $pb_2()$ *compute single backward steps 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.*

Proof. The correctness of $pb_1()$ and $pb_2()$ is a direct consequence of Lemmas 2 and 3. Moreover, by the conditions of Procedure 1, the set of minimal pushout complements (possibly restricted to $\mathcal{Q}$) is finite and computable. Thus, $pb_1()$ and $pb_2()$ are effective. □

4. Well-quasi-orders for graph transformation systems

In this section we prove the compatibility of three different orders with our general framework: the minor ordering, the subgraph ordering and the induced subgraph ordering. These orders can all form Q-restricted WSTSs and differ in their expressiveness.

4.1. Minor ordering

We first instantiate the general framework with the minor ordering. Robertson and Seymour showed in a seminal result that the minor ordering is a wqo on the set of all graphs [6], even for hypergraphs [7]. We will extend their results to our minor ordering, a slight variant of their definition, and apply case (i) of Theorem 1 to obtain decidability results.

Definition 12 *(Minor).* A graph G_1 is a *minor* of a graph G_2 (written $G_1 \sqsubseteq G_2$), if G_1 can be obtained from G_2 by a sequence of deletions of nodes (including all incident edges) and contractions of edges. An *edge contraction* deletes an edge, chooses an arbitrary equivalence relation on the nodes incident to the edge and merges all nodes in each equivalence class. (This includes edge deletion as a special case when all equivalence classes are singletons.)

We call a partial morphism $\mu : G_1 \rightharpoonup G_2$ a *minor morphism* (written $\mu : G_1 \mapsto G_2$) if it is surjective, injective on edges and, whenever $\mu(v) = \mu(w) = z$ for some $v, w \in V_{G_1}$ and $z \in V_{G_2}$, there exists an undirected path between v and w in G_1 where all nodes on the path are mapped to z and μ is undefined on every edge on the path.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Lemma 4. *The minor ordering is representable by minor morphisms (cf. Definition 8).*

Proof. We first show that minor morphisms are closed under composition. Let $\mu\colon G_1 \mapsto G_2$, $\mu'\colon G_2 \mapsto G_3$ be two minor morphisms. Obviously the composition $\mu' \circ \mu$ is surjective and injective on edges. It remains to show that the third property is also satisfied.

So let v, w be two nodes of G_1 with $\mu'(\mu(v)) = \mu'(\mu(w)) = z$. Since μ' is a minor morphism there exists a path $v'_0, e'_1, v'_1, \ldots, e'_n, v'_n$ between $\mu(v)$ and $\mu(w)$ in G_2 (i.e. $\mu(v) = v'_0$ and $\mu(w) = v'_n$), where e'_i is incident (among others) to nodes v'_{i-1}, v'_i. Furthermore all nodes v'_i are mapped to z by μ' and the image of all edges is undefined.

The morphism μ is surjective, hence there exist edges $e_1, \ldots, e_n$ in G_1 such that $\mu(e_i) = e'_i$, where e_i is incident to nodes v_{i-1}, w_i (i.e. v_{i-1}, e_i, w_i is a path) with $\mu(v_i) = v'_i = \mu(w_i)$. Since μ is a minor morphism, v_i and w_i (for $0 < i < n$) are connected by a path $x^i_0, f^i_1, x^i_1, \ldots, f^i_{m_i}, x^i_{m_i}$ in G_1 where $w_i = x^i_0$ and $v_i = x^i_{m_i}$. Furthermore, since $\mu(v_0) = v'_0 = \mu(v)$ there exists a path $x^0_0, f^0_1, x^0_1, \ldots, f^0_{m_0}, x^0_{m_0}$ from $v = x^0_0$ to $v_0 = x^0_{m_0}$ and analogously a path $x^n_0, f^n_1, x^n_1, \ldots, f^n_{m_n}, x^n_{m_n}$ from $w_n = x^n_0$ to $w = x^n_{m_n}$. Also, the image of all these edges under μ is undefined. So the combined path

$$x^0_0, f^0_1, \ldots, f^0_{m_0}, x^0_{m_0}, e_1, x^1_0, f^1_1, \ldots, f^{n-1}_{m_{n-1}}, x^{n-1}_{m_{n-1}}, e_n, x^n_0, f^n_1, \ldots, f^n_{m_n}, x^n_{m_n}$$

connects v and w and it satisfies all the requirements of Definition 12.

We still need to show that a minor morphisms $\mu : G_2 \mapsto G_1$ exists if and only if $G_1 \sqsubseteq G_2$. Assume $G_1 \sqsubseteq G_2$, then G_1 can be obtained from G_2 by deleting nodes and contracting edges as specified in Definition 12. Clearly each of these operations can be separately specified by a minor morphism. If such operations are applied repeatedly, the result follows from the fact that minor morphisms are closed under composition.

Conversely, let $\mu : G_2 \mapsto G_1$ be a minor morphism. Now perform the following operations on G_2. First, determine all nodes in G_1 which have more than one preimage under μ. Since all preimages have to be connected by paths in G_2, where μ is undefined on the edges in the paths, we can contract all such edges, resulting in a graph G' with a minor morphism $G_2 \mapsto G'$, where all nodes with the same image (under μ) have been merged. Afterwards, if an edge in G_2 has no image under μ, then we can delete it from G_2. If a node v has no image under μ, then, since μ is a morphism, it is clear that all edges incident to v also do not have an image under μ. Hence, we can delete these edges from G, leaving us with an isolated node, which can then be deleted. Continuing this process we will obtain G_1, and since we have restricted ourselves only to the allowed operations, it is clear that G_1 is a minor of G_2. □

Corollary 1. *(See [7].) The minor ordering is a wqo on the class of all graphs.*

Proof. We can obtain this result as a corollary of the results in [7], since the notion of collapse is very similar to the notion of minor morphisms. However, we have to translate between the two settings. In Proposition 1.6 of [7] Robertson and Seymour use hypergraphs where nodes are ordered with respect to an edge (as in our case), but where edges can only be incident to the same node once, i.e., the sequence of nodes incident to an edge does not contain duplicates. This difference can be remedied by replacing edges which are incident to some node more than once with new edges of lower arity (and a new label). Thus, we assume that our edges are incident to a disjoint sequence of nodes. Furthermore, our set of labels is well-quasi-ordered (as required by Robertson and Seymour) by the identity, since it is finite.

We now consider the sequence $G_1, G_2, G_3, \ldots$ and show the existence of a minor morphism between two of these graphs. According to Proposition 1.6 of [7] there exist indices $i < j$ such that there is a collapse of G_j to G_i. More precisely, there exists a function η with domain $V_{G_i} \cup E_{G_i}$ such that:

1. $\eta(v)$ is a non-empty connected subgraph of $K_{V_{G_j}}$ (where K_V is the undirected complete graph on the node set V) and the graphs $\eta(u), \eta(v)$ are pairwise disjoint for distinct $u, v \in V_{G_i}$.
2. $\eta(e) \in E_{G_j}$ for all $e \in E_{G_i}$ and η is injective on edges and label-preserving.
3. For $e \in E_{G_i}$ if $c_{G_i}(e) = v_1 \ldots v_n$, then $c_{G_j}(\eta(e)) = u_1 \ldots u_n$ and u_i is contained in the subgraph $\eta(v_i)$ for every $i \in \{1, \ldots, n\}$.
4. For each $v \in V_{G_i}$ and each (undirected) edge f in $\eta(v)$, connecting $x, y \in V_{G_j}$, there exists an edge $e \in E_{G_j}$ which is adjacent to x, y. Furthermore e is not in the image of η. (The latter can be assumed since our label alphabet is finite and each label is associated with an arity. Hence every edge is bounded, i.e., has a finite neighborhood.)

Now define a minor morphism $\mu\colon G_j \mapsto G_i$ as follows:

- An edge e' of G_j is mapped to e in G_i whenever $\eta(e) = e'$. If no such edge exists $\mu(e)$ is undefined. This is well-defined since η is injective on edges (Condition 2). Furthermore μ is injective and surjective on edges and preserves labels.
- Whenever a node v' of G_j is contained in a subgraph $\eta(v)$ we map v' to v. Otherwise $\mu(v')$ is undefined. Clearly due to Condition 1 μ obtained in this way is well-defined and surjective on nodes.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

We next verify that μ is a partial morphism. Assume that $\mu(e') = e$ with $c_{G_i}(e) = v_1 \dots v_n$ and $c_{G_j}(e') = u_1 \dots u_n$: then $\eta(e) = e'$ and u_i is contained in $\eta(v_i)$ (Condition 3). Hence u_i is mapped to v_i, which means that the image of all nodes is defined and the map μ is structure-preserving.

Finally assume that $\mu(v') = \mu(w') = z$. This means that v', w' are both contained in $\eta(z)$. Since the subgraph $\eta(z)$ is connected there exists a path from v' to w' in $\eta(z)$. Let us denote this path by $v' = v'_0, f_1, v'_1, \dots, v'_{n-1}, f_n, v'_n$. By Condition 4 we can require that there exist edges $e'_k \in E_{G_j}$, which are not in the image of η and adjacent to v'_{k-1}, v'_k. This implies the existence of a path $v'_0, e'_1, v'_1, \dots, v'_{n-1}, e'_n, v'_n$ such that $\mu(e'_k)$ is undefined and $\mu(v'_k) = z$ (since all nodes $v'_0, \dots, v'_n$ are within the subgraph $\eta(z)$ and are hence mapped to z).

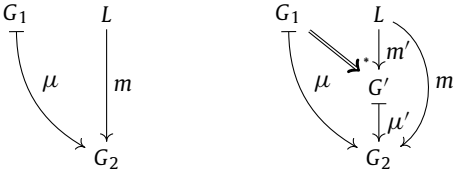
This means that $\mu\colon G_j \mapsto G_i$ is a minor morphism. Note that the collapse relation of [7] is finer than the minor ordering of Definition 12. Especially a minor morphism might map straight edges to loops, which is not allowed in the collapse. However, this only means that we might "miss" some pairs of related graphs, but we will always find one. □

There are different classes of graph transformation systems which are well-structured wrt. the minor ordering. For instance context-free graph grammars, where the left-hand side of every rule consists of a single hyperedge, connected to a sequence of distinct nodes. Whenever $G_1 \sqsubseteq G_2$ and G_1 contains such a single hyperedge as a left-hand side, then G_2 will contain it as well, proving that the rule is also applicable to the larger graph. This argument can be extended to every GTS, where left-hand sides of rules consist of disconnected edges and no nodes are deleted. Moreover, any GTS can be transformed into a WSTS by adding edge contraction rules for every label (rules deleting an edge and merging its nodes according to some partition). In fact, it is sufficient to add contraction rules for edges of arity larger than one. These rules can be used to rewrite a graph to a minor to make rules applicable, but affect the transition system induced by the GTS. In this way we obtain some form of lossy system and it has been observed before [4] that it is often possible to make a system well-structured by introducing lossiness.

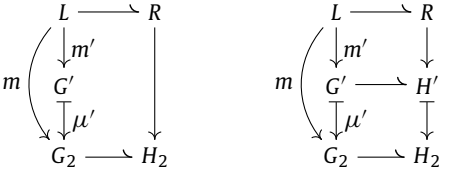
Proposition 3 *(WSTSs wrt. the minor ordering). Let $\mathcal{R}$ be a GTS that contains edge contraction rules for every edge label of arity larger than one. More concretely, for $\ell \in \Lambda$ with $ar(\ell) > 1$ we require rules that delete this edge and contract the incident nodes according to every possible partition, disregarding only the trivial partition where each node is only equivalent to itself.*

Then $\mathcal{R}$ is a WSTS.

Proof. Assume that G_2 is a minor of G_1, witnessed by a minor morphism μ, and there exists a total match $m\colon L \to G_2$ of rule $r\colon L \rightharpoonup R$ (see diagram below on the left).



Then, by mimicking the edge contractions performed by μ (but not the edge and node deletions), we can rewrite G_1 to G' via the contraction rules in $\mathcal{R}$ and we obtain the situation above on the right. Note that G_2 is a subgraph of G' and that m' is conflict-free whenever m is conflict-free.



The pushout square can be split into two pushouts via standard pushout splitting (see diagram on the right) and from a variation of Lemma 5 it follows that the resulting morphism $H' \mapsto H_2$ is a minor morphism. Since μ' does not contract any edges, the deletions performed by μ' are either done by the rule as well (dangling edges) or can be performed in order to obtain H_2 from H'. Indeed, H_2 is again a subgraph of H'. Hence G_1 can first be rewritten to G' and then G' is rewritten in one step to H', with $H_2 \sqsubseteq H'$. □

With the following two lemmas we show that the conditions for soundness and completeness of our backward steps are satisfied by minor morphisms. An example for preservation along total morphisms is shown in Fig. 6. Intuitively, every deletion and contraction of an x by μ can be performed by μ' as well, by deleting or contracting $g(x)$ (since g is total). Even if g is non-injective, contractions are always possible.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

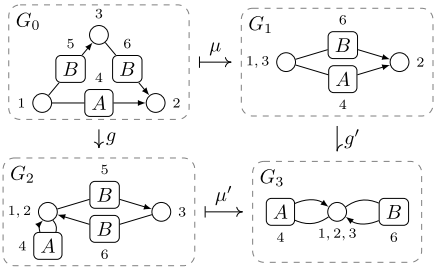


Fig. 6. Minor morphisms are preserved by pushouts along total morphisms.

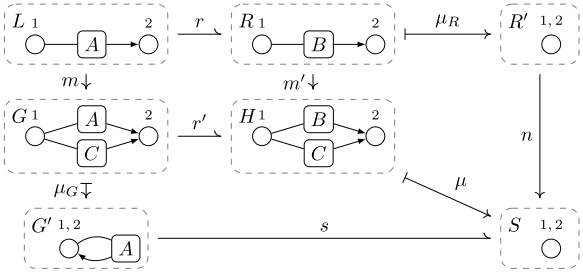


Fig. 7. Minor morphisms are pushout closed.

Lemma 5. *Minor morphisms are preserved by pushouts along total morphisms (cf. Definition 9).*

Proof. Let G_3 be the pushout of G_0 along $\mu : G_0 \mapsto G_1$ and $g : G_0 \to G_2$. Let $v, w \in V_{G_2}$ be two nodes that are mapped to the same node $z \in V_{G_3}$ via the morphism $\mu' : G_2 \rightharpoonup G_3$. But this means that v and w are in the same equivalence class, and thus necessarily have pre-images v' and w' in G_0.

Now, since they are in the same equivalence class, there exists a sequence of nodes $y_1, y_2, \ldots, y_n \in V_{G_0}$ with $y_1 = v'$ and $y_n = w'$ such that $\mu(y_1) = \mu(y_2), g(y_2) = g(y_3), \ldots$ Since μ is a minor morphism there exists a path (in G_0) from y_i to y_{i+1}, with $i \in \{1, 3, \ldots\}$ such that all nodes on the path are mapped to $\mu(y_i)$. Since g is total, there also exists a path from $g(y_i)$ to $g(y_{i+1})$, with $i \in \{1, 3, \ldots\}$ in G_2. Now, due to commutativity, all nodes on such a path (in G_2) will be mapped to the same node in G_3. Also due to commutativity, the images of all the edges in this path are undefined, since the equivalence class is not valid (due to μ being a minor morphism). Further, since $g(y_i) = g(y_{i+1})$ for $i \in \{2, 4, \ldots\}$, there is a path from $g(y_1) = v$ to $g(y_n) = w$ such that all nodes on that path are mapped to the same node, $z \in V_{G_3}$, and none of the edges in this path lie in the domain of μ'. Also, surjectivity (on nodes and edges) and injectivity (on edges) is preserved by the pushout construction.

Thus, μ' is a minor morphism. □

An example for pushout closure is shown in Fig. 7. Given L, R, G, H, S and the corresponding morphisms, we can find minor morphisms μ_R and μ_G such that S is the pushout of $\mu_R \circ r$, $\mu_G \circ m$ by simulating μ. Since μ contracts both the B-edge and C-edge, we contract the B-edge in μ_R and the C-edge in μ_G (since it has no preimage in R). This ensures that both edges are contracted when forming the pushout S.

Lemma 6. *Minor morphisms are pushout closed (cf. Definition 10).*

Proof. We will prove that minor morphisms are pushout closed by first constructing R', G' together with minor morphisms $\mu_R : R \mapsto R'$ and $\mu_G : G \mapsto G'$. We then prove that S is the pushout of $\mu_R \circ r$, $\mu_G \circ m$ and that $\mu_G \circ m$ is conflict-free wrt. r.

Construction of R' and μ_R From R, construct a minor R' (and simultaneously a minor morphism μ_R) as follows: first, let R' be simply a copy of R. For $e \in E_R$, if the image of e in H under m' is contracted to construct S, then contract the corresponding edge in R'. In this case e is undefined under μ_R and its incident nodes are mapped to merged nodes in R'. If e is deleted (without contracting the nodes), delete it in R' as well, and leave e undefined under μ_R. Now, let $v \in V_R$ be such that its image in H is deleted in constructing S. This implies that the image of v in H is either an isolated node, or all its incident edges were deleted. So since we deleted corresponding edges in R', we can safely delete v in R', and leave it undefined under μ_R. (Note that it is not possible for R to have an edge that is not mapped to an edge in H, since m is total and conflict-free and hence m' must be total).

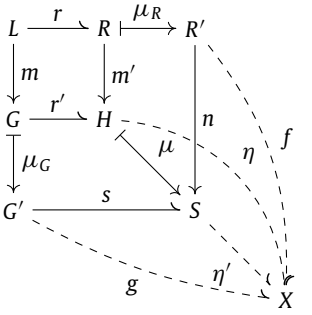
Now, R' is a minor of R, because the construction involved only the allowed operations. Further, due to its construction, μ_R is a minor morphism.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Construction of G' and μ_G Perform a similar construction for G', with one difference: For $x \in G$ such that x has a pre-image in L, do not contract/delete it in G', even if x had an image in H that was contracted/deleted in constructing S. (The intuition for this is that it is enough for an item to be contracted/deleted by one of the minor morphisms for it to be contracted/deleted in the pushout graph and a second contraction/deletion would result in a non-conflict-free match.) The rest of the construction is as before. Again, G' is a minor of G and $\mu_G : G \mapsto G'$ is a minor morphism. Further $\mu_G \circ m$ is total since m is total and μ_G is defined for all elements with a pre-image in L.

Conflict-freeness of $\mu_G \circ m$ Also, $\mu_G \circ m$ is conflict-free with respect to r. To see this, suppose there exist elements $x_1, x_2 \in L$ such that $(\mu_G \circ m)(x_1) = (\mu_G \circ m)(x_2)$. Whenever x_1, x_2 are edges, then $m(x_1) = m(x_2)$, since μ_G does not merge edges. In this case by assumption r is either undefined on both or defined on both. A similar argument applies whenever x_1, x_2 are nodes and $m(x_1) = m(x_2)$. So now assume that x_1, x_2 are nodes and $y_1 = m(x_1) \neq m(x_2) = y_2$. Then, $\mu_G(y_1) = \mu_G(y_2)$ implies either $r(x_1) = r(x_2)$ or y_1 and y_2 have distinct images in H with a path connecting them which is contracted while constructing S. Hence, $r(x_1)$ and $r(x_2)$ are both defined and distinct. Thus there cannot be a deletion/preservation conflict.

Construction of n Now, we construct the morphisms $n : R' \to S$ and $s : G' \rightharpoonup S$ (see diagram on the right). For any $x \in R$ we define n as $n(\mu_R(x)) = \mu(m'(x))$. To see that this is valid, first note that μ_R is surjective and if $\mu_R(x)$ is undefined, then $\mu(m'(x))$ will also be undefined because of the construction of μ_R. Then, if there exist x_1, x_2 such that $\mu_R(x_1) = \mu_R(x_2)$, then x_1 and x_2 must be nodes and not edges, because μ_R is injective on edges. And we must have $\mu(m'(x_1)) = \mu(m'(x_2))$ since μ_R only contracts if μ contracts. Thus n is well-defined.



Further, m' is total (since m is total and conflict-free), $\mu(m'(x))$ is undefined iff μ is undefined on $m'(x)$, and in that case $\mu_R(x)$ will also be undefined. Combined with the fact that μ_R is surjective, this implies that n is total. Also, the relevant part of the diagram commutes, due to the definition of n.

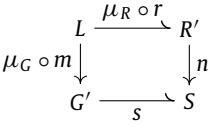
Construction of s Similarly, s is defined as $s(\mu_G(x)) = \mu(r'(x))$. By essentially the same argument as in the case of n, we can show that this definition is valid and the relevant part of the above diagram commutes. However, in this case s may be partial, because for a node x with a pre-image in L, $r'(\mu(x))$ may be undefined but $\mu_G(x)$ will still be defined. For such a node, s will be left undefined.

Commutativity and correctness of the outer square Furthermore it is straightforward to shown that n, s satisfy all properties of a morphism. And finally, since each of the parts of the above diagram commutes, the diagram as a whole also commutes.

Existence of the mediating morphism η' Now, let there be some graph X and two morphisms $f : R' \rightharpoonup X$ and $g : G' \rightharpoonup X$, such that $(f \circ \mu_R) \circ r = (g \circ \mu_G) \circ m$, then, since H is a pushout, there exists $\eta : H \rightharpoonup X$ such that η is the unique morphism with $f \circ \mu_R = \eta \circ m'$ and $g \circ \mu_G = \eta \circ r'$. For $x \in H$, define a morphism $\eta' : S \to X$ as follows $\eta'(\mu(x)) = \eta(x)$. Since μ is surjective, every element $y \in S$ has a pre-image $x \in H$, hence η' can in principle be defined for every such y by the above definition. The current situation is depicted in the diagram above.

Correctness and uniqueness of η' It is left to show that η' is well-defined and unique. Now, if there exist x_1, x_2 in H such that $\mu(x_1) = \mu(x_2)$, then x_1 and x_2 must be nodes (since μ is a minor morphism), and further, there must be a path connecting them, such that all nodes on this path are also mapped to the same node. Then, if $\eta(x_1) \neq \eta(x_2)$, there exist y_1, y_2 which lie on this path from x_1 to x_2, such that they are adjacent, and $\mu(y_1) = \mu(y_2)$ but $\eta(y_1) \neq \eta(y_2)$ (this includes the case where $\eta(y_1)$ is defined and $\eta(y_2)$ undefined, or vice versa). Let e be the edge connecting them. It must have a pre-image in either R or G (or both). Suppose e has a pre-image e' in R with the pre-images of y_1 and y_2 being y_1' and y_2' respectively. Then, if e is contracted in S it holds that e' is contracted in R', and hence $\mu_R(y_1') = \mu_R(y_2')$. But then, $f \circ \mu_R = \eta \circ m'$ implies that $\eta(y_1) = \eta(y_2)$, which leads us to a contradiction. Now, suppose e has a pre-image e'' in G instead of R. If e'' does not have a pre-image in L, then we arrive at a contradiction by a similar argument as before. On the other hand, if e'' has a pre-image in L, then e must have a pre-image in R (since H is a pushout), hence the previous argument applies. Hence, such x_1, x_2 cannot exist, and η' is well-defined.

This gives us an $\eta' : S \rightharpoonup X$ such that $\eta' \circ \mu = \eta$. This implies $f \circ \mu_R = (\eta \circ \mu) \circ m'$ and $(g \circ \mu_G) = (\eta \circ \mu) \circ r'$. Now η and therefore $\eta' \circ \mu$ is the *unique* morphism with this property. Since μ is fixed and surjective, this means that η' is the unique morphism such that $f = n \circ \eta'$ and $g = s \circ \eta'$. Thus, the diagram to the right is a pushout. As shown before, n and $\mu_G \circ m$ are both total, and $\mu_G \circ m$ is conflict-free with respect to r. □



Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Before stating the decidability result for the minor ordering we need to consider the question of how pushout complements can be constructed when some (but not all) of the morphisms involved may be partial. Consider a diagram $L \overset{r}{\rightharpoonup} R \overset{n}{\to} S$ and let $dom(r)$ be the subgraph of L consisting of all nodes and edges on which r is defined.

$$\begin{array}{ccccc} L & \xrightarrow{r_1} & dom(r) & \xrightarrow{r_2} & R \\ {\scriptstyle m}\downarrow & & {\scriptstyle \tilde{m}}\downarrow & & {\scriptstyle n}\downarrow \\ X & \xrightarrow{\tilde{r}_1} & \tilde{X} & \xrightarrow{\tilde{r}_2} & S \end{array}$$

The idea is to split $L \overset{r}{\rightharpoonup} R$ into $L \overset{r_1}{\rightharpoonup} dom(r) \overset{r_2}{\to} R$, as shown above, where r_2 is total and r_1 is an inverse injection, i.e., a morphism which is injective, surjective, but not necessarily total. Since it is well-known that every pushout can be split into two pushouts (as shown above), the task of computing pushout complements can be divided into two subtasks, the computation of $\tilde{X}$ and the computation of X, given $\tilde{X}$.

Lemma 7. *Let L and R be graphs, $r_1 : L \rightharpoonup \tilde{L}$ be an inverse injection, and $\tilde{m} : \tilde{L} \to \tilde{X}$ be a total morphism. Now construct a specific pushout complement X with morphisms $m : L \to X$, $\tilde{r}_1 : X \rightharpoonup \tilde{X}$ as follows:*

1. *Take a copy of the graph $\tilde{X}$, and let m be $\tilde{m} \circ r_1$. The morphism $\tilde{r}_1$ is the identity.*
2. *Let Y be the set of elements of L the image of which is undefined under r_1. Add a copy of Y to this copy of $\tilde{X}$ (more specifically: take the pushout of r_1^{-1} and $\tilde{m}$), and extend m by mapping Y into this set. Furthermore $\tilde{r}_1$ 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[2] equivalences. The morphisms m and $\tilde{r}_1$ 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 $r_1, \tilde{m}$ and the corresponding morphisms $m : L \to X$ are total. Any other pushout complement X' where $m' : L \to X'$ is total (as shown on the right) has some graph $X \in \mathcal{P}$ as a minor.

$$\begin{array}{ccc} L & \xrightarrow{r_1} & \tilde{L} \\ \downarrow{\scriptstyle m'} & & \downarrow{\scriptstyle \tilde{m}} \\ X' & \xrightarrow{\tilde{r}'_1} & \tilde{X} \end{array}$$

Furthermore, if m' is conflict-free wrt. to r_1, then there exists a pushout complement $X \in \mathcal{P}$ where m is conflict-free wrt. r_1, such that $X \sqsubseteq X'$.

Proof. First, it is clear that the graphs produced by the above construction will in fact be pushout complements. Also, m will be total.

Let X' be any pushout complement with m' total. Since r_1 is an inverse injection and m', $\tilde{m}$ are total, $\tilde{r}'_1$ must also be an inverse injection. So the entire graph $\tilde{X}$ has a pre-image in X', or we can say that X' contains an "exact copy" of $\tilde{X}$.

Now, if v is a node in X', then either it belongs to the copy of $\tilde{X}$, or if not, it has a pre-image in Y under m'. Similarly for an edge e in X', either e is in the copy of $\tilde{X}$, or it has at least one endpoint with a pre-image in L.

This means that X' must be of the following form: it must contain an exact copy of $\tilde{X}$, and images of all elements in Y (since m' must be total). An element in the copy of $\tilde{X}$ cannot have a preimage in Y under m' since otherwise $\tilde{m}$ would not be total. But the elements of Y may be merged with each other in any fashion. It cannot contain any other nodes, but it may contain any number of additional edges, so long as at least one endpoint of these edges lies outside the copy of $\tilde{X}$.

For any such X', we can delete these extra edges, and we will obtain a minor X of X', which is still a pushout complement. Since this minor contains only a copy of $\tilde{X}$ and extra elements from Y, it can be obtained by the above construction. Thus the above construction allows us to compute all the minimal pushout complements.

Finally, whenever m' is conflict-free wrt. r_1, we can find a matching X for which $m : L \to X$ is also conflict-free, since the minor morphism $X' \mapsto X$ only deletes edges, but never contracts them. So m' must be conflict-free whenever m is. □

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

Proposition 4. *Let L, R and S be graphs, with a partial morphism $r : L \rightharpoonup R$ and a total morphism $n : R \to S$. Then, if we apply the following procedure we construct pushout complements X of r, n and any other pushout complement X' (with $m' : L \to X$ where m' is total and conflict-free wrt. r) has one of them as a minor.*

1. *Split r into two morphisms as follows: let $r_1 : L \rightharpoonup dom(r)$ be an inverse injection and let $r_2 : dom(r) \to R$ be total.*
2. *Now construct all pushout complement of r_2 and n as usual for total morphisms. (See [14] for a detailed description of this step.)*
3. *Let $\tilde{X}$ be any such pushout complement with $\tilde{m} : dom(r) \to \tilde{X}$.*

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

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

4. *For $r_1, \tilde{m}$ use the construction of Lemma 7 in order to obtain the minimal pushout complements X (with total and conflict-free m).*
5. *Finally, from all such pushout complements X take the minimal ones.*

The situation is depicted in the diagram below.

$$\begin{array}{ccccc} L & \xrightarrow{r_1} & dom(r) & \xrightarrow{r_2} & R \\ \downarrow m & & \downarrow \tilde{m} & & \downarrow n \\ X & \longrightarrow & \tilde{X} & \longrightarrow & S \end{array}$$

Proof. Since both squares in the diagram above are pushouts (the left square by Lemma 7, the right square by construction), the outer square must be a pushout and hence X is a pushout complement of r, n.

Now take any pushout complement X' such that $m' : L \to X'$ is total and conflict-free wrt. r. The situation is as depicted below on the left:

$$\begin{array}{ccccc} L & \xrightarrow{r_1} & dom(r) & \xrightarrow{r_2} & R \\ \downarrow m & & & & \downarrow n \\ X & & \longrightarrow & & S \end{array} \qquad \begin{array}{ccccc} L & \xrightarrow{r_1} & dom(r) & \xrightarrow{r_2} & R \\ \downarrow m & & \downarrow \tilde{m} & & \downarrow n \\ X & \longrightarrow & \tilde{X} & \longrightarrow & S \end{array}$$

As shown above on the right we can now split the pushout leading to two pushout diagrams, where $\tilde{X}$ is one of the pushout complements of r_2, n computed above. Hence by Lemma 7 the graph X' must have one of the constructed graphs X as a minor. □

Proposition 5. *The coverability problem wrt. the minor ordering is decidable for every GTS satisfying the conditions of Proposition 3.*

Proof. Every GTS satisfying the conditions of Proposition 3 is a WSTS and therefore also a $\mathcal{Q}$-restricted WSTS, where $\mathcal{Q}$ is the class of all graphs. Due to Lemmas 4, 5 and 6 as well as Proposition 1, the necessary correctness criteria of Procedure 1 (Variant 2) are satisfied. Furthermore, according to Proposition 4 the set of minimal pushout complements is computable, such that all conditions of the procedure are satisfied. The decidability results of Theorem 1 case (i) apply. □

(a) A process generates a new message to elect itself as leader

(b) Other processes forward a message if their ID is higher than that of the sender

(c) A process receiving its own message is the leader

(d) A process P_i or leader L leaves the ring

Fig. 8. Modeling of a leader election protocol by graph transformation rules.

Example 2. This decidability result can now be used to obtain a verification algorithm for systems modeled by graph transformation, especially for lossy systems. Let for instance the graph transformation system in Fig. 8 be given which models a leader election protocol on a ring structure (see Fig. 10). Any process can generate a message (Fig. 8a) to acquire leadership. These messages are passed along the ring (Fig. 8b), but only if the passing process has a lower priority than the process who generated the message. If a message reaches the process who generated it (Fig. 8c), the process is elected as leader. Processes and leaders may also decide to leave the ring (Fig. 8d). Note that due to the presence of contraction rules for edges of arity two and larger, these rules satisfy the restrictions of Proposition 3, guaranteeing that Procedure 1 will return the correct results.

An error occurs in the system if two different processes are elected as leaders. This can be modeled by the error graph in Fig. 9, which also represents any larger ring with two leaders. Starting with this graph our procedure will compute the set of graphs shown in Fig. 11, representing all possible predecessor graphs. None of these graphs is a minor of the initial configuration (Fig. 10), proving that the error cannot occur in the modeled system.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

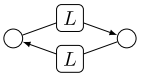


Fig. 9. Error configuration of the leader election protocol.

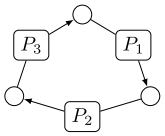


Fig. 10. Initial ring structure of the leader election protocol, where every process has a unique ID.

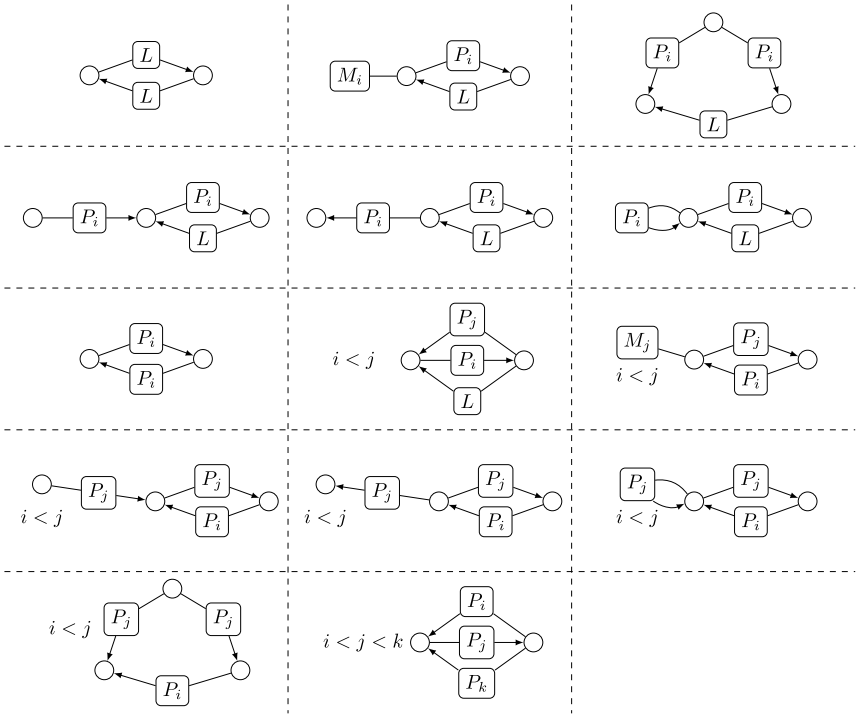


Fig. 11. Final working set computed by Procedure 1 for the leader election protocol.

For the verification in our tool UNCOVER we have to fix the largest process ID, e.g. three as in Fig. 10, but the results in Fig. 11 are generic, for arbitrarily high IDs.

4.2. Subgraph ordering

The second order presented in this paper is the subgraph ordering. It satisfies the conditions of Procedure 1 for a restricted set of graphs and is therefore also compatible with our framework. We already stated a related result (but for injective instead of conflict-free matches) in [10], but did not yet instantiate a general framework.

Definition 13 *(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 $\mu : G \rightarrowtail S$ a *subgraph morphism* if and only if it is injective on all elements on which it is defined and surjective.

Lemma 8. *The subgraph ordering is representable by subgraph morphisms (cf. Definition 8).*

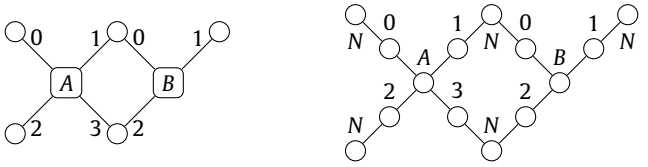
Using a result from Ding [8] we can show that the set $\mathcal{G}_k$ of hypergraphs where the length of every undirected path is bounded by a constant k, is well-quasi-ordered by the subgraph relation. A similar result was shown by Meyer for depth-bounded systems in [16]. Note that we bound undirected path lengths instead of directed path lengths. For the class

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

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).

Proposition 6. *The subgraph ordering is a wqo on the class $\mathcal{G}_k$ for every k.*

Proof. In [8] Ding showed that this proposition holds for undirected, simple graphs with node labels. Thus, we only need to give an encoding of hypergraph into undirected graphs which does not arbitrarily increase the length of paths. A suitable encoding can be seen below. Note that in the hypergraphs the label of an edge determines its arity and hence there is an upper bound on the numbers used as node labels.



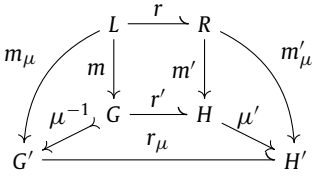
For each node in the hypergraph we have an N-labeled node in the undirected graph and for each X-labeled edge in the hypergraph we have an X-labeled node in the undirected graph. The connections between hyperedges and nodes are modeled by adding additional nodes labeled with their position in the incident node sequence. These nodes are then connected to (via undirected edges) the nodes added for the corresponding edge and incident node. It is worth pointing out that every path in the hypergraph still exists in the undirected graph and no new paths are added (which do not correspond to paths in the hypergraph). However, a path in the undirected graph need not start and end with an N-labeled node, but it might contain up to three additional nodes at the beginning and end. Furthermore four edges correspond to one hyperedge. Thus, if the longest path in the hypergraph is bounded by k, then the longest path in the undirected graph is bounded by $4k+6$ (a longer path would imply a path of length $k+1$ in the hypergraph).

Any infinite sequence of hypergraphs $G_0, G_1, \ldots$ can be translated to a sequence of undirected graphs $G'_0, G'_1, \ldots$. According to Ding, if the longest path is bounded, indices $i < j$ exists, such that $G'_i \subseteq G'_j$, which implies $G_i \subseteq G_j$. □

We can show that every GTS satisfies the compatibility condition of Definition 2 naturally wrt. the subgraph ordering and obtain the following result.

Proposition 7 *(WSTSs 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.*

Proof. We have to show that whenever $G \Rightarrow H$ and $G \subseteq G'$, then there exists H' with $G' \Rightarrow^* H'$ (here even $G' \Rightarrow H'$) and $G' \subseteq H'$.



Let $r : L \rightharpoonup R$ be a rule and $m : L \to G$ a matching that is conflict-free wrt. r such that G is rewritten to H, i.e. the upper inner square on the right is a pushout. Furthermore let $\mu : G' \rightarrowtail G$ be a subgraph morphism, then the inverse morphism μ^{-1} is total and injective. Thus, the morphism $m_\mu : L \to G'$ with $m_\mu = \mu^{-1} \circ m$ in the diagram on the right is total and conflict-free wrt. r and G' can be rewritten to H' as indicated by r_μ.

Since the outer square is a pushout, there is a unique morphism $\mu' : H \to H'$ such that the lower inner square commutes. Furthermore since the upper inner square and the outer square are pushouts, so is the lower inner square. This means that μ' is total and injective since μ^{-1} is and pushouts preserve this properties. Hence, $H \subseteq H'$.

Since the compatibility condition is naturally satisfied and due to Proposition 6 the subgraph ordering is a wqo on $\mathcal{G}_k$, we have proven the proposition. □

Lemma 9. *Subgraph morphisms are preserved by pushouts along total morphisms (cf. Definition 9).*

Proof. Let G_3 be the pushout of G_0 along $\mu : G_0 \rightarrowtail G_1$ and $g : G_0 \to G_2$. Since every subgraph morphism is also a minor morphism, we can use Lemma 5 to prove this lemma. It remains to be shown that $\mu' : G_2 \rightarrowtail G_3$ is injective for all vertices and edges for which it is defined.

Assume there are two different elements $x_1, x_2 \in G_2$ such that $\mu'(x_1) = \mu'(x_2)$. For G_3 to be a pushout, both x_1 and x_2 have to have preimages $x'_1, x'_2 \in G_0$ with $g(x'_1) = x_1$ and $g(x'_2) = x_2$. The diagram commutes, thus μ is defined for both elements and these elements are mapped injectively (since mu is a subgraph morphism) to $x''_1, x''_2 \in G_1$ respectively. Hence, there is a commuting diagram with $g'(x''_1) = \mu'(x_1) \neq g'(x''_2) = \mu'(x_2)$, for which there is no mediating morphism from G_3. Since this violates the pushout properties of the diagram, μ' has to be injective. □

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Lemma 10. *Subgraph morphisms are pushout closed (cf. Definition 10).*

Proof. This lemma immediately follows from Lemma 6. Note that by construction μ_R and μ_G only contract edges if μ does, thus, μ_R and μ_G are subgraph morphisms. □

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 8. *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.*

Proof. In Lemmas 8, 9 and 10 we have shown that the subgraph ordering satisfies the conditions of Procedure 1. Furthermore, the set of minimal pushout complements – not just restricted to $\mathcal{G}_k$ – can be computed in the same way as it is done for the minor ordering, such that both variants of Procedure 1 are applicable. □

The existence of a pred-basis leads to another interesting result. Given a GTS, a graph G and a constant k, the question whether from G we can reach a graph which contains a path of length at least k, is decidable.

Proposition 9. *Let $\mathcal{R}$ be a finite set of rules, let G_0 be a graph and let k be a constant. Furthermore, let $(\mathcal{G}, \Rightarrow)$ be the transition system on all graphs generated by $\mathcal{R}$. The following problem is decidable: is there a graph G with $G_0 \Rightarrow^* G$ and G has a path of length at least k?*

Proof. We use that paths are preserved by the subgraph ordering, i.e. if a graph G has a path of length k, then any larger graph also has a path of length k (or longer). We observe that the problem above is therefore equivalent to checking if a path of length k is coverable or not and we will now show that this is decidable.

We first need to find a finite set $\mathcal{P}_k$ of graphs such that its upward-closure is the class of all graphs which contain a path of length k or longer. For directed graphs this set simply consists of all paths of length k, where labels and directions of edges are arbitrary. However, for hypergraphs $\mathcal{P}_k$ is more complex and we define it as the set of all connected graphs which contain a path of length k and are minimal in the sense that any deletion of a node or edge decreases every longest path of the graph by at least one. This set $\mathcal{P}_k$ represents every graph F containing a path of length k (or longer), since the subgraph of F consisting only of its k-path, including all nodes incident to the paths edges, is an element of $\mathcal{P}_k$. Note that every graph of $\mathcal{P}_k$ is by definition connected, contains exactly k edges and its number of nodes is bounded by the edges arities. Thus there are only finitely many possibilities for such graphs, i.e. $\mathcal{P}_k$ is finite.

Let $\mathcal{W}$ be the result of applying Variant 1 of Procedure 1 to $\mathcal{P}_k$ using the given rule set and restricting the graph class to $\mathcal{G}_k$. Then $G \in \uparrow\mathcal{W}$ implies that G can cover a path of $\mathcal{P}_k$ and $G \notin \uparrow\mathcal{W}$ implies that G cannot cover a path of $\mathcal{P}_k$ in $\Rightarrow_{\mathcal{G}_k}$. However, any minimal pushout complement not in $\mathcal{G}_k$ and therefore not added in Step 2 would have been subsumed in Step 3, since every such pushout complement is already represented by the initial working set. Thus, the restriction to $\mathcal{G}_k$ has no effect on the result and $G \notin \uparrow\mathcal{W}$ in fact implies that G cannot cover a path of $\mathcal{P}_k$ in $\Rightarrow$. □

Due to this proposition we can check whether it is sufficient to use Variant 1 of Procedure 1 to compute a final working set $\mathcal{W}$ that faithfully represents *all* predecessors of a given set of error configurations $\mathcal{F}$. According to case (ii) of Theorem 1, if only graphs with paths bounded by k are reachable from some graph G, then G covers an error *if and only if* $G \in \uparrow\mathcal{W}$.

However the restricted coverability problem is undecidable even for $\mathcal{G}_k$. We can show this by a simple reduction from the reachability problem for two-counter machines. 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 10. *Let $k \geq 2$ be a natural number. The restricted coverability problem for $\mathcal{G}_k$-restricted well-structured GTSs with the subgraph ordering is undecidable.*

Proof. For this reduction we use the control state reachability problem of Minsky machines which is undecidable since Minsky machine can simulate Turing machines [17]. We reduce this problem to the restricted coverability problem using the subgraph ordering on the set of graphs $\mathcal{G}_2$, where the length of the longest undirected path is less than or equal to two. Let $(Q, \Delta, (q_0, m, n))$ be the Minsky machine, where Q is the set of states, $\Delta \subseteq Q \times Cmd \times Q$ is the set of instructions and (q_0, m, n) defines the initial state and counter values. A command of Cmd can be an increment, a decrement or a zero-test, as shown in Fig. 13. We define a GTS using $\{q, q^B \mid q \in Q\} \cup \{c_1, c_2, X\}$ as the set of labels. The initial graph is shown in Fig. 12 and illustrates how configurations of the Minsky machine are represented as graphs.

For each transition rule of the Minsky machine, we add a graph transformation rule as shown in Fig. 13. A counter is represented as a star-like structure with a main node as center, where the value of the counter is the number of attached

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

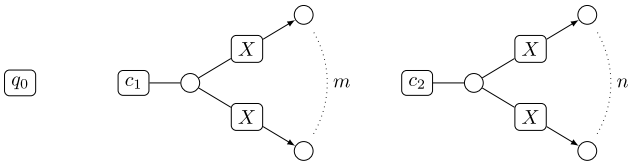


Fig. 12. The initial configuration of the Minsky machine represented by a graph.

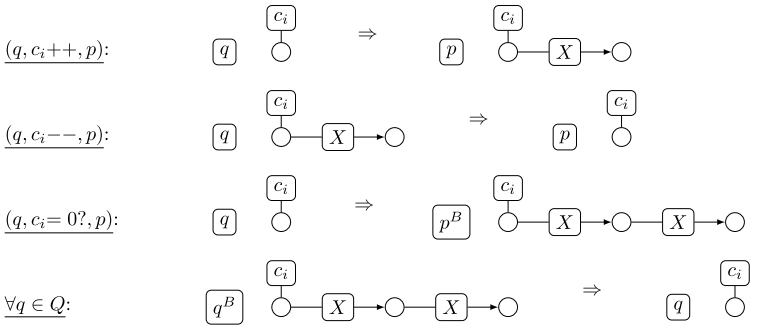


Fig. 13. Translation of Minsky rules to GTS rules.

X-edges. The center nodes are marked by unary c_1- and c_2-edges respectively. Incrementing and decrementing corresponds to creating and deleting X-edges incident to the appropriate center node. Regardless of the counters value, the longest undirected path of this structure has at most length two.

The zero-test adds two X-edges and blocks the state-edge, such that the rewritten graph has an undirected path of length three if and only if the counter was not zero (i.e. had an X-edge attached). The auxiliary rules unblock the state to enable further computation.

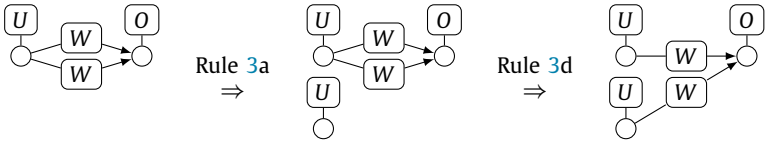
Obviously, if there is a sequence of transitions of the Minsky machine which leads from a configuration (q_0, m, n) to a state q_f, this sequence can be copied in the GTS and every graph generated through this sequence is in $\mathcal{G}_2$. On the other hand, if the graph consisting of a single q_f-edge is $\mathcal{G}_2$-restricted coverable in the GTS, there is a sequence of rule applications corresponding to a sequence of transitions of the Minsky machine. Since this rule applications generate only graphs in $\mathcal{G}_2$, the zero-test-rule is only applied if the counters value is in fact zero and the sequence of transitions is valid.

Instead of adding and removing a path of length two in the last two rules of Fig. 13 one can add and remove a path of length k to show the undecidability for $\mathcal{G}_k$-restricted well-structured GTSs. □

Example 3. 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 8 to calculate all graphs which lead to some error configuration.

For instance, let a multi-user system as described in Example 1 be given. Usually, 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 Fig. 4 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 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 Fig. 3d mapping both nodes 2 and 3 to the right node.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

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 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, so that we also have to bound the multiplicity of edges between two nodes. Note that this restriction is implicitly done in [8] 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 modeling, 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 can equip the rules with a limited form of negative application conditions, still retaining the compatibility condition of Definition 2.

Definition 14 *(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.

Lemma 11. *The induced subgraph ordering is presentable by induced subgraph morphisms (cf. Definition 8).*

We can again transfer Ding's results for the induced subgraph ordering to our setting. However, we have to explicitly bound the number of edges between two nodes to obtain a wqo. In Ding's proof this restriction is done implicitly, since parallel edges are forbidden in general.

Proposition 11. *Let n, k be natural numbers. The induced subgraph ordering is a wqo on the set of directed, edge-labeled 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).*

Proof. We prove this proposition by induction over the type of a graph, adapting Ding's proof in [8] that undirected, node-labeled graphs of bounded type are well-quasi-ordered by the induced subgraph order.

A graph which consists of at most a single node with possibly incident edges (loops) has type one. A connected graph containing at least two nodes has at most type n, if there is a node v so that the deletion of v and all attached edges splits the graph into components which have each type at most $n-1$. The type of a non-connected graph is the maximal type of its components. As already shown by Ding, every directed graph, where the longest undirected path has length n, has at most type $n+2$. Note that contrary to Ding our type is bounded by $n+2$ instead of n, because we measure path lengths via the number of edges instead of nodes and Ding excludes paths of length n to obtain graphs of type at most n.

To prove this proposition we use hypergraphs which are additionally node labeled, i.e. there is a second alphabet Σ of node labels and a (total) labeling function $\sigma : V_G \to \Sigma$. We obtain classical directed graphs if $|\Sigma| = 1$.

Let $G_1, G_2, \ldots$ be an infinite sequence of graphs of type n and with edge multiplicity bounded by k. If $n = 1$ then every G_i consists of a single node with up to $k \cdot |\Lambda|$ attached loops. Since the sets of node and edge labels are finite, there are only finitely many possibilities to attach up to $k \cdot |\Lambda|$ edges to the node, thus $G_i \trianglelefteq G_j$ for some $i < j$, i.e. $\trianglelefteq$ is a wqo on the set of all such graphs.

Now let $n > 1$. Then there is a node $v_i \in G_i$ such that the deletion of v_i (and its attached edges) splits the graph into components $G_{i,q}$ (for $1 \le q \le \ell_i$) of type at most $n-1$. We define $\widetilde{G}_i$ to be the graph containing only v_i and its attached loops. Additionally we define $\widehat{G}_{i,q}$ to be $G_{i,q}$ where the label $\sigma(y)$ of every node y is changed to $\sigma'(y) = (f_y, \sigma(y))$, where $f_y : \Lambda \to \{0, 1, \ldots, k\}^2$ is a function such that $f_y(\lambda) = (a, b)$ where a is the number of incoming and b of outgoing λ-labeled edges attached to both y and v_i. Since there are only finitely many possible functions f_y (due to the multiplicity constraint), the set of labels remains finite. We extend $\trianglelefteq$ to sequences such that $(\widetilde{G}_i, \widehat{G}_{i,1}, \ldots, \widehat{G}_{i,\ell_i}) \trianglelefteq^* (\widetilde{G}_j, \widehat{G}_{j,1}, \ldots, \widehat{G}_{j,\ell_j})$ if and only if $\widetilde{G}_i \trianglelefteq \widetilde{G}_j$ and there are $p_1, \ldots, p_{\ell_i}$ with $1 \le p_1 < \ldots < p_{\ell_i} \le \ell_j$ such that $\widehat{G}_{i,q} \trianglelefteq \widehat{G}_{j,p_q}$. As shown for the case $n = 1$, $\trianglelefteq$ is a wqo on all $\widetilde{G}_i$ and since the graphs $\widehat{G}_{i,q}$, $\widehat{G}_{j,p_q}$ are of type $n-1$, they are well-quasi-ordered by induction hypothesis. Hence, due to Higman [9] $\trianglelefteq^*$ is also a wqo and there are indices $i < j$ such that $(\widetilde{G}_i, \widehat{G}_{i,1}, \ldots, \widehat{G}_{i,\ell_i}) \trianglelefteq^* (\widetilde{G}_j, \widehat{G}_{j,1}, \ldots, \widehat{G}_{j,\ell_j})$.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

It remains to be shown that this implies $G_i \trianglelefteq G_j$. By Lemma 11 there are induced subgraph morphisms $\mu_0 : \widetilde{G}_j \rightarrowtail \widetilde{G}_i$ and $\mu_q : \widehat{G}_{j,p_q} \rightarrowtail \widehat{G}_{i,q}$ for $1 \leq q \leq \ell_i$. We define the morphism $\mu : G_j \to G_i$ as

$$\mu(x) = \begin{cases} v_i & \text{if } x = v_j \\ \mu_q(x) & \text{if } x \in \widehat{G}_{j,p_q} \text{ for some } q \\ \mu_0(x) & \text{if } x \in E_{\widetilde{G}_j} \text{ and } c_{\widetilde{G}_j}(x) = v_j v_j \\ \mu^v(x) & \text{if } x \in E_{G_j} \text{ and } c_{G_j}(x) = v_j v \vee c_{G_j}(x) = v v_j \\ & \text{for } v_j \neq v \in V_{G_j} \text{ and } \mu(v) \text{ is defined} \\ \text{undefined} & \text{else} \end{cases}$$

where μ^v is any total, bijective morphism from G_j – restricted to v_j, v and the edges between them – to G_i – restricted to v_i, $\mu_q(v)$ (if $v \in \widehat{G}_{j,p_q}$) and any edges between them (both not including loops). Note that μ^v exists since v and $\mu_q(v)$ are labeled with some (f, α), thus the number of edges between v_j and v is equal to the number of edges between v_i and $\mu_q(v)$ for all labels and directions.

We now show that μ is a induced subgraph morphism. First note that μ is a valid morphism since μ_q, μ_0 and μ^v are morphisms and labels of edges in G_i, G_j are the same as their representative in $\widehat{G}_{j,p_q}$, $\widehat{G}_{i,q}$ and representatives of nodes are labeled with (f, α), where the original is labeled α also implying equality on labels. We then observe that μ is injective and surjective, since μ_q, μ_0 and μ^v are all injective and surjective and v_j is mapped to v_i. Assume there is an edge $e \in E_{G_j}$ for which μ is undefined. If e is contained in one of the components $\widehat{G}_{j,p_q}$ or in $\widetilde{G}_j$, at least one attached node is undefined, since μ_q and μ_0 are an induced subgraph morphisms. If e connects v_j and a node v of a component $\widehat{G}_{j,z}$, then either z is not of the form p_q and μ is undefined on $\widehat{G}_{j,z}$ or $z = p_q$ and $\mu(v)$ is undefined since otherwise μ^v has a mapping for e. Since μ is an induced subgraph morphism, we obtain that $G_i \trianglelefteq G_j$. □

Proposition 12 *(WSTSs wrt. the induced subgraph ordering). Let n, k be natural numbers and let $\mathcal{G}_{n,k}$ be a set of directed, edge-labeled 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.*

Proof. We first show that every GTS satisfies the compatibility conditions by modifying the proof of Proposition 7 by additionally showing that the reverse of μ' is an induced subgraph morphism. Assume there is an edge $e \in E_{H'}$, where all attached nodes have a preimage under μ' but e has none. Since r', μ^{-1}, μ', r_μ is a pushout, this can only be the case if all nodes attached to e have a preimage in G and G' and e has a preimage in G'. Because μ is an induced subgraph morphism, e has a preimage in G. Due to commutativity r' cannot be undefined on this preimage, thus, e has to have a preimage in H, violating the assumption. As shown in Proposition 11, the induced subgraph ordering is a wqo on $\mathcal{G}_{n,k}$, which concludes this proof. □

Lemma 12. *Induced subgraph morphisms are preserved by pushouts along total morphisms (cf. Definition 9).*

Proof. Since every induced subgraph morphism is also a subgraph morphism, μ' is injective and surjective by Lemma 9. Let $e \in E_{G_2}$ be an edge on which μ' is undefined. e has a preimage $e' \in G_0$ since otherwise the pushout of μ and g would contain e. Since $\mu'(g(e'))$ is undefined, so is $g'(\mu(e'))$. In fact μ is undefined for e' because otherwise g and μ would be defined on e and e would be in the pushout. μ is an induced subgraph morphism, thus at least one of the nodes v attached to e' is undefined and also μ' has to be undefined on $g(v)$ for the diagram to commute. □

Lemma 13. *Induced subgraph morphisms are pushout closed (cf. Definition 10).*

Proof. In Lemma 10 we have shown that there are subgraph morphisms μ_R and μ_G if μ is a subgraph morphism. We will show that these morphisms are induced subgraph morphisms if μ is an induced subgraph morphism.

Let $e \in E_R$ be an edge on which μ_R is undefined. By definition $\mu(m'(e))$ is undefined and since m' is total, μ is undefined on $m'(e)$ (which is defined). Hence, at least one node v attached to $m'(e)$ has no image under μ and all its preimages under m' (which exist since e has a preimage) are undefined under μ_R. Thus, e is attached to at least one node on which μ_R is undefined on.

Let $e' \in E_G$ be an edge on which μ_G is undefined. By definition $\mu(r'(e'))$ is undefined and e' has no preimage under m. Because of the latter property, e' is in the pushout H and therefore defined under r'. Thus, μ is undefined on $r'(e)$ and on at least one attached node. All preimages under r' of this node are undefined under μ_G since the diagram commutes. Hence, μ_G is an induced subgraph morphism. □

Proposition 13. *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.*

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Proof. As shown in Lemmas 11, 12 and 13 the induced subgraph ordering satisfies the conditions of Procedure 1. The computation of minimal pushout complements is more involved than in the subgraph case. This is due to the fact that if a rule deletes a node, all attached edges are deleted, even if these edges have no preimage in L. Adding an edge to a pushout complement and attaching it to a node which is deleted by the rule, results in another pushout complement. Contrary to the subgraph ordering these pushout complements are not already represented by the graph without the edge, if the induced subgraph ordering is used, but we can compute them as follows:

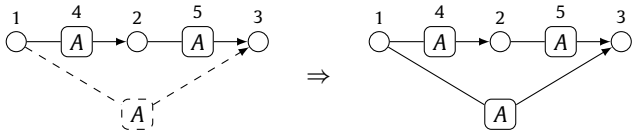
Let $(r : L \rightharpoonup R) \in \mathcal{R}'$ be a rule and $m : R \to G$ a match calculated in Step 1 of Procedure 1. Calculate the set of minimal pushout complements $\mathcal{G}_{poc}$ wrt. the subgraph ordering restricted to $\mathcal{G}_{n,k}$. For all pushout complement objects $X \in \mathcal{G}_{poc}$ with morphism $r' : X \rightharpoonup G$, add all X' to $\mathcal{G}_{poc}$, where X' can be obtained by adding an edge to X, which is attached to at least one node on which r' is undefined. Do not add X' if it exceeds the bounded multiplicity. Perform the previous step until $\mathcal{G}_{poc}$ becomes stationary, which will be the case since the multiplicity is bounded. The set $\mathcal{G}_{poc}$ is then the set of minimal pushout complement objects wrt. the induced subgraph ordering. □

The computation of minimal pushout complements in this case is considerably more complex, since extra edges have to be added (see the proof of Proposition 13), but we also obtain additional expressiveness in two different senses. On the one hand, there are classes of graphs which are upward closed wrt. the induced subgraph ordering, but not wrt. the subgraph ordering, enabling us to express properties not expressible with subgraphs (see Example 4). On the other hand, there are GTSs with negative application conditions which are still well-structured wrt. the induced subgraph ordering (see Example 5), although negative application conditions violate the compatibility condition wrt. the subgraph ordering.

Example 4. We use the graph transformation system in Example 1 in connection with the induced subgraph ordering. Note that the unary edges can be replaced by loops to obtain directed graphs. One of the properties which can be verified using the induced subgraph ordering is that every node is incident to a loop, defining its type as user or object. This can be done by using a single node without any edges as error graph. The node represents all graphs, where there is a node which has no loops attached to it. By applying the backward algorithm we obtain only graphs which also have this property, implying that this property is invariant with respect to the graph transformation system. Since a node without a loop is a subgraph of a node with a loop, this property is not expressible by the subgraph ordering.

In general negative application conditions violate the compatibility condition wrt. the subgraph ordering, since rules may become inapplicable to a graph by adding nodes or edges. Restricted GTSs may still be well-structured even with negative application conditions, as we have shown in [18] for the minor ordering. However, since the induced subgraph ordering is finer, there are graph transformation systems with negative application conditions, which satisfy the compatibility condition naturally (wrt. the induced subgraph ordering).

Example 5. 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 match only for the solid part of the left-hand side and this match cannot be extended to a match for the solid and dashed part.



Applied to a graph with only A-edges, this rule calculates the transitive closure and will finally terminate. 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. Hence 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 removing all obtained graphs which do not satisfy one of the negative application conditions.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Table 1
Runtime result for different case studies.

case study	wqo	class $\mathcal{Q}$	var.	runtime	#EG
Leader election ($id \leq 10$)	minor	all graphs	1 / 2	1 m 2 s	451
Leader election ($id \leq 20$)	minor	all graphs	1 / 2	27 m 56 s	2401
Termination det. (faulty)	minor	all graphs	1 / 2	796 ms	69
Termination det. (correct)	minor	all graphs	1 / 2	304 ms	101
Rights management	subg.	all graphs	2	28 ms	4
Dining Philosophers	subg.	all graphs	2	442 ms	12
Public–private server	subg.	path ≤ 50	1	14.1 s	104
Public–private server	subg.	path ≤ 100	1	3 m 28 s	204

4.4. Implementation

We implemented Procedure 1 with support for the minor ordering as well as the subgraph ordering in the tool UNCOVER (available via www.ti.inf.uni-due.de/research/tools/uncover/). The tool is written in C++ and designed in a modular way for easy extension with further orders. One of the few 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 the leader election protocol in Fig. 8 (with a variable number of processes) 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 Fig. 3 as well as a public–private server protocol, using the subgraph order. In addition the table contains a variant of the dining philosophers problem on an arbitrary graph structure modeled in [19]. 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 (#EG). Note that the minimal representatives shown in Fig. 11 are a compact representation of the graphs computed for each bound on the process ID ($id \leq 10, 20$). Despite this bound the system graph might still be of arbitrary size, since there can be several processes with the same ID and unboundedly many messages can be generated.

5. Conclusion

We have presented a general framework for viewing GTSs as ($\mathcal{Q}$-)restricted WSTSs and discussed three important instantiations, using the minor ordering, the subgraph ordering and the induced subgraph ordering. Furthermore we presented two case studies, a leader election protocol and the management of read and write access rights. Finally, we discussed our implementation with very encouraging runtime results.

We already introduced an extension of the presented framework with rules, which can uniformly change the entire neighborhood of nodes in [19]. 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 [18] 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 [20] in order to see whether they can be integrated into this framework and to study application scenarios.

Another promising direction for future work is to enrich graphs with constraints (similar to [21]), in order to compute symbolic representations of graphs with attributes. This would enable us to automatically generate the graphs with constraints of Fig. 11. So far we have computed them manually and the tool requires an upper bound on the process Ids.

Instead of considering only backward steps, it might also be promising to conduct a forward analysis similar to [22]. Furthermore it would be interesting to obtain exact complexity bounds for our analysis. We expect the complexity to be non-elementary and to be placed somewhere in the extended Grzegorczyk hierarchy [23].

Related work In the last few years, several contributions emerged which view structure or graph transformation as a WSTS. Some of this work (such as [24]) can also be summarized under the name *parameterized verification*, which attempts to show the correctness of a protocol or algorithm for arbitrary, unspecified communication structures.

We now review some closely related papers: in [22] the authors use the subgraph ordering and a forward search to prove fair termination for depth-bounded systems. In [25] 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 [24] 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 neighborhood of arbitrary size. The work in [26] uses a backwards search on graph patterns in order to verify an ad-hoc routing protocol, but not in the setting of WSTSs. Finally, [27] considers complex decidability questions in the spirit of our Proposition 8: “is it decidable whether only graphs/processes of type X are reachable”?

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005

Acknowledgments

We would like to thank Roland Meyer, for giving us the idea to consider the subgraph ordering, and Giorgio Delzanno for several interesting discussions on wqos and WSTSs. A very special thanks goes to Salil Joshi who was involved in the beginning of this line of research, but could not join us in writing this paper.

References

[1] B. König, J. Stückrath, A general framework for well-structured graph transformation systems, in: Proc. of CONCUR '14, in: LNCS/ARCoSS, vol. 8704, Springer, 2014, pp. 467–481.
[2] S. Joshi, B. König, Applying the graph minor theorem to the verification of graph transformation systems, in: Proc. of CAV '08, in: LNCS, vol. 5123, Springer, 2008, pp. 214–226.
[3] P.A. Abdulla, K. Čerāns, B. Jonsson, Y. Tsay, General decidability theorems for infinite-state systems, in: Proc. of LICS '96, IEEE, 1996, pp. 313–321.
[4] A. Finkel, P. Schnoebelen, Well-structured transition systems everywhere!, Theor. Comput. Sci. 256 (1–2) (2001) 63–92.
[5] G. Rozenberg (Ed.), Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1: Foundations, World Scientific, 1997.
[6] N. Robertson, P. Seymour, Graph minors. XX. Wagner's conjecture, J. Comb. Theory, Ser. B 92 (2) (2004) 325–357.
[7] N. Robertson, P. Seymour, Graph minors XXIII. Nash–Williams' immersion conjecture, J. Comb. Theory, Ser. B 100 (2010) 181–205.
[8] G. Ding, Subgraphs and well-quasi-ordering, J. Graph Theory 16 (1992) 489–502.
[9] G. Higman, Ordering by divisibility in abstract algebras, Proc. Lond. Math. Soc. s3-2 (1) (1952) 326–336.
[10] N. Bertrand, G. Delzanno, B. König, A. Sangnier, J. Stückrath, On the decidability status of reachability and coverability in graph transformation systems, in: Proc. of RTA '12, in: LIPIcs, vol. 15, Schloss Dagstuhl – Leibniz Center for Informatics, 2012, pp. 101–116.
[11] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer, Fundamentals of Algebraic Graph Transformation, Monographs in Theoretical Computer Science, Springer, 2006.
[12] H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, A. Corradini, Algebraic approaches to graph transformation—part II: Single pushout approach and comparison with double pushout approach, in: Rozenberg [5], Ch. 4, pp. 247–312.
[13] S. Joshi, B. König, Applying the graph minor theorem to the verification of graph transformation systems, Tech. rep. 2012-01, Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, 2012.
[14] M. Heumüller, S. Joshi, B. König, J. Stückrath, Construction of pushout complements in the category of hypergraphs, in: Selected Revised Papers from the Workshop on Graph Computation Models, GCM 2010, in: Electronic Communications of the EASST, vol. 39, 2011.
[15] M. Koch, L. Mancini, 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, in: LNCS, vol. 2502, Springer, 2002, pp. 229–243.
[16] R. Meyer, Structural stationarity in the π-calculus, Ph.D. thesis, Carl-von-Ossietzky-Universität Oldenburg, 2009.
[17] M.L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, Inc., 1967.
[18] B. König, J. Stückrath, Well-structured graph transformation systems with negative application conditions, in: Proc. of ICGT '12, in: LNCS, vol. 7562, Springer, 2012, pp. 89–95.
[19] G. Delzanno, J. Stückrath, Parameterized verification of graph transformation systems with whole neighborhood operations, in: Proc. of RP '14 (Reachability Problems), in: LNCS, vol. 8762, Springer, 2014, pp. 72–84.
[20] M. Fellows, D. Hermelin, F. Rosamond, Well-quasi-orders in subclasses of bounded treewidth graphs, in: Proc. of IWPEC '09 (Parameterized and Exact Computation), in: LNCS, vol. 5917, Springer, 2009, pp. 149–160.
[21] F. Orejas, Symbolic graphs for attributed graph constraints, J. Symb. Comput. 46 (3) (2011) 294–315.
[22] K. Bansal, E. Koskinen, T. Wies, D. Zufferey, Structural counter abstraction, in: Proc. of TACAS '13, in: LNCS, vol. 7795, Springer, 2013, pp. 62–77.
[23] S. Schmitz, P. Schnoebelen, The power of well-structured systems, in: Proc. of CONCUR '13, in: LNCS, vol. 8052, Springer, 2013, pp. 5–24.
[24] G. Delzanno, A. Sangnier, G. Zavattaro, Parameterized verification of ad hoc networks, in: Proc. CONCUR '10, in: LNCS, vol. 6269, Springer, 2010, pp. 313–327.
[25] P.A. Abdulla, A. Bouajjani, J. Cederberg, F. Haziza, A. Rezine, Monotonic abstraction for programs with dynamic memory heaps, in: Proc. of CAV '08, in: LNCS, vol. 5123, Springer, 2008, pp. 341–354.
[26] M. Saksena, O. Wibling, B. Jonsson, Graph grammar modeling and verification of ad hoc routing protocols, in: Proc. of TACAS '08, in: LNCS, vol. 4963, Springer, 2008, pp. 18–32.
[27] R. Hüchting, R. Majumdar, R. Meyer, Bounds on mobility, in: Proc. of CONCUR '14, in: LNCS/ARCoSS, vol. 8704, Springer, 2014, pp. 357–371.

Please cite this article in press as: B. König, J. Stückrath, Well-structured graph transformation systems, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.03.005