Parameterized Verification of Graph Transformation Systems with Whole Neighbourhood Operations*

Giorgio Delzanno[1] and Jan Stückrath[2]

[1] Univerità di Genova, Italy
giorgio.delzanno@unige.it
[2] Universität Duisburg-Essen, Germany
jan.stueckrath@uni-due.de

Abstract. We introduce a new class of graph transformation systems in which rewrite rules can be guarded by universally quantified conditions on the neighbourhood of nodes. These conditions are defined via special graph patterns which may be transformed by the rule as well. For the new class for graph rewrite rules, we provide a symbolic procedure working on minimal representations of upward closed sets of configurations. We prove correctness and effectiveness of the procedure by a categorical presentation of rewrite rules as well as the involved order, and using results for well-structured transition systems. We apply the resulting procedure to the analysis of the Distributed Dining Philosophers protocol on an arbitrary network structure.

1 Introduction

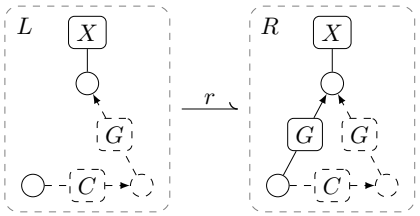
Parameterized verification of distributed algorithms is a very challenging task. Distributed algorithms are often sensible to the network topology and they are based on communication patterns like broadcast messages and conditions on channels that can easily generate undecidable verification instances or finite-state problems of high combinatorial complexity. In order to naturally model interaction rules of topology-sensitive protocols it seems natural to consider languages based on graph rewriting and transformations as proposed in [21]. However, in this formalism rules can only match fixed subgraph in the graph they are applied to. Since we need to specify rules where the entire neighbourhood of a node is matched by the rule, we extend the standard approach by universally quantified patterns attached to nodes. With these patterns the matching of a left side of a rule can be increased until the entire neighbourhood of a node is covered. If the matching cannot be extended in this way the rule is not applicable, e.g. we could formalize a rule which only matches a node when every incident edge is incoming. Additionally the matched occurrences of the patterns can also be changed by the rule. A similar approach are adaptive star grammars [20], the difference being that we do not restrict our left rule sides to be stars.

The resulting formal language can be applied to specify distributed versions of concurrent algorithms like Dining Philosophers in which neighbour processes use channels to request and grant access to a given shared resource. The protocol we use has been

* Research partially supported by DFG project GaReV.

proposed by Namjoshi and Trefler in [25]. There requests are specified using process identifiers attached to edges representing point-to-point communication channels. Universally quantified guards are used to ensure mutual exclusive access to a resource. In this paper we formulate the protocol without need of introducing identifiers. We instead use our extended notion of graph transformation systems to specify ownership of a given communication link. Universally quantified patterns attached to a requesting node are used then as guards to ensure exclusive access. Erroneous or undesirable configurations in the algorithm can be presented by a set of minimal error configurations. We then use a backward procedure to check if a configuration containing one of the error configurations is reachable. If none is reachable, the algorithm is proven to be correct.

Following the approach proposed in [7,24], we use basic ingredients of graph transformation and category theory (e.g. pushouts) to formally specify the operational semantics of our model. Parameterized verification for the resulting model is undecidable in general, even without universally quantified patterns [7]. To overcome this problem, we provide an approximated symbolic backward procedure using result for well-structured transition systems [6,22] to guarantee correctness and termination. Although the over-approximation is based on the monotonic abstraction approach proposed in [3,5], its application to the considered class of infinite-state systems is highly non trivial. In fact, our universal quantification approach is not restricted to process states only, but it can specify complex graph patterns as shown on the right. There the node marked with the X-edge represents a group where every node attached with a G-edge is a member of. The rule can be applied if every edge attached to the two solid nodes is matched and has the form of the dashed part (the quantification). Effectively the rule adds a node to a group if all other connected nodes (via a C-edge) are already members of the group.



We have implemented a prototype version of the algorithms in the tool UNCOVER and tested on some case-studies. For instance, our prototype can verify the Distributed Dining Philosophers example without need of additional invariants as in [25]. Due to space limitations, the proofs can be found in an extended version of this paper [17].

2 Preliminaries

In this paper we use hypergraphs, a generalization of directed graphs, where an edge can connect an arbitrary large but finite set of nodes. Furthermore we use graph morphisms to define rewriting rules.

Hypergraph Let Λ be a finite sets of edge labels and $ar\colon \Lambda \to \mathbb{N}$ a function that assigns an arity to each label (including the arity zero). A *(Λ-)hypergraph* (or simply graph) is a tuple (V_G, E_G, c_G, l_G) where V_G is a finite set of nodes, E_G is a finite set of edges,

$c_G\colon E_G \to V_G^*$ is a connection function and $l_G\colon E_G \to \Lambda$ is an edge labelling function. We require that $|c_G(e)| = ar(l_G(e))$ for each edge $e \in E_G$. An edge e is called *incident* to a node v if v occurs in $c_G(e)$. An *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.

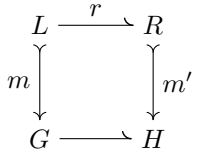
Let G, G' be (Λ-)hypergraphs. A *partial hypergraph morphism* (or simply *morphism*) $\varphi\colon G \rightharpoonup G'$ consists of a pair of partial functions $(\varphi_V : V_G \rightharpoonup V_{G'}, \varphi_E : E_G \rightharpoonup E_{G'})$ such that for every $e \in E_G$ it holds that $l_G(e) = l_{G'}(\varphi_E(e))$ and $\varphi_V(c_G(e)) = c_{G'}(\varphi_E(e))$ whenever $\varphi_E(e)$ is defined. Furthermore if a morphism is defined on an edge, it must be defined on all nodes incident to it. We denote *total morphisms* by an arrow of the form $\to$ and write $\rightarrowtail$ if the total morphism is known to be injective.

Pushout Our rewriting formalism is the so-called *single-pushout approach* (SPO) based on the categorical notion of pushouts in the category of graphs and partial graph morphisms [21]. Given two morphisms $\varphi : G_0 \rightharpoonup G_1$ and $\psi : G_0 \rightharpoonup G_2$, the *pushout* of φ, ψ consists of the graph G_3 and two morphisms $\varphi' : G_2 \rightharpoonup G_3$ and $\psi' : G_1 \rightharpoonup G_3$. It corresponds to a merge of G_1 and G_2 along a common interface G_0 while at the same time deleting every element of one of the graphs if it has a preimage in G_0 which is not mapped to an element in the other graph. It is known that in our category the pushout of two morphisms always exists and is unique (up to isomorphism). It can be computed in the following way.

Let $\equiv_V$ and $\equiv_E$ be the smallest equivalences on $V_{G_1} \cup V_{G_2}$ and $E_{G_1} \cup E_{G_2}$ satisfying $\varphi(v) \equiv_V \psi(v)$ for all $v \in V_{G_0}$ and $\varphi(e) \equiv_E \psi(e)$ for all $e \in E_{G_0}$. The nodes and edges of the pushout object G_3 are then all *valid* equivalence classes of $\equiv_V$ and $\equiv_E$. An equivalence class is *valid* if it does not contain the image of some $x \in G_0$ for which $\varphi(x)$ or $\psi(x)$ is undefined. The equivalence class of an edge is also considered invalid if it is incident to a node with an invalid equivalence class. The morphisms φ' and ψ' map each element to its equivalence class if this class is valid and are undefined otherwise.

For a backward step in our procedure we also need the notion of a *pushout complement* which is, given $\varphi : G_0 \rightharpoonup G_1$ and $\psi' : G_1 \rightharpoonup G_3$, a graph G_2 and morphisms $\psi : G_0 \rightharpoonup G_2$, $\varphi' : G_2 \rightharpoonup G_3$ such that G_3 is the pushout of φ, ψ. For graphs pushout complements not necessarily exist and if they exist there may be infinitely many. See [23] for a detailed description on how pushout complements can be computed.

GTS A *rewriting rule* is a partial morphism $r\colon L \rightharpoonup R$, where L is called left-hand and R right-hand side. A *match* (of r) is a total and injective morphism $m : L \rightarrowtail G$. Given a rule and a match, a *rewriting step* or rule application is given by a pushout diagram as shown on the right, resulting in the graph H. Note that injective matchings are not a restriction since non-injective matchings can be simulated, but are necessary for universally the quantified rules defined later.



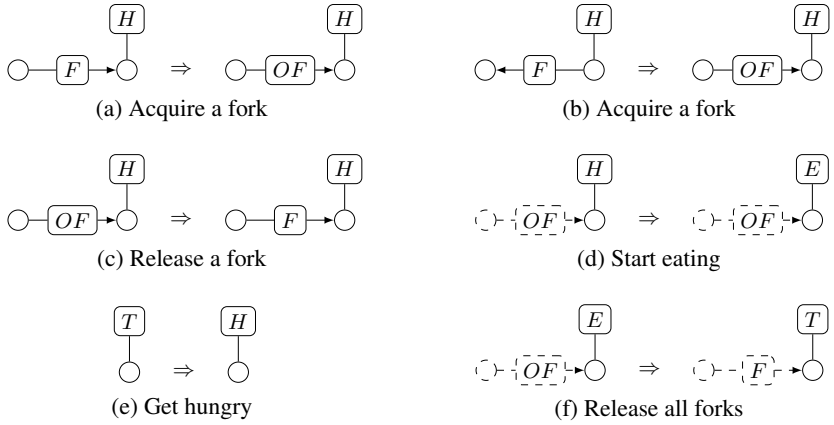
A *graph transformation system (GTS)* is a finite set of rules $\mathcal{R}$. Given a fixed set of graphs $\mathcal{G}$, a *graph transition system* on $\mathcal{G}$ generated by a graph

transformation system $\mathcal{R}$ is represented by a tuple $(\mathcal{G}, \Rightarrow)$ where $\mathcal{G}$ is the set of states and $G \Rightarrow G'$ if and only if $G, G' \in \mathcal{G}$ and G can be rewritten to G' using a rule of $\mathcal{R}$.

A computation is a sequence of graphs $G_0, G_1, \ldots$ s.t. $G_i \Rightarrow G_{i+1}$ for $i \geq 0$. G_0 can reach G_1 if there exists a computation from G_0 to G_1.

3 Graph Transformations with Universally Quantified Conditions

To clarify the ideas and illustrate the usefulness of universally quantified conditions on the neighbourhood of nodes, let us consider the following example.



(a) Acquire a fork (b) Acquire a fork

(c) Release a fork (d) Start eating

(e) Get hungry (f) Release all forks

Fig. 1: Modelling of the dining philosophers problem on an arbitrary net

Example 1. Figure 1 shows a set of rules describing the Dining Philosophers Problem on an arbitrary graph structure. Each node represents a philosopher who can be in one of three different states: hungry (H), eating (E) or thinking (T). Each state is indicated by a unary edge attached to the philosopher. Between two philosophers there may be a free fork (an F-edge) or a fork owned by one of the philosophers (an OF-edge pointing to its owner). Note that our directed edges are in fact hyperedges of arity two, where the first node is the source and the second node is the target.

Philosophers can take unowned forks (Figure 1a and 1b) and also release control (Figure 1c). If a philosopher owns all connected forks, he can start to eat (Figure 1d). The dashed part of the rule indicates a universal quantification, meaning that the rule can only be applied if all edges attached to the philosopher are part of the matching and in fact forks owned by him. At some point the philosopher finished eating, releasing all forks (Figure 1f) and may become hungry in the future (Figure 1e). When releasing all forks, all forks owned by the philosopher are converted to unowned forks.

Rules matching the entire neighbourhood of a node (in the following called *quantified node*), such as the rules in Figure 1d and 1f cannot be described by normal rewriting

rules. Therefore we extend normal rules to so-called universally quantified rules consisting of a normal rule and a set of universal quantifications. The idea is to first find a matching for the rule and then extend the rule as well as the matching until the entire neighbourhood of quantified nodes is part of the matching.

We apply the rule in Figure 1f to the graph G shown in Figure 2. There exists a match $m : L \rightarrowtail G$ where $r : L \rightharpoonup R$ is the rule without any use of the quantification. However, this matching does not match the entire neighbourhood of the quantified node (marked grey). Before applying the rule we have to add multiple copies of the quantification to r generating a so-called instantiation η where the extended match $\overline{m}$ contains the entire neighbourhood of the quantified node.

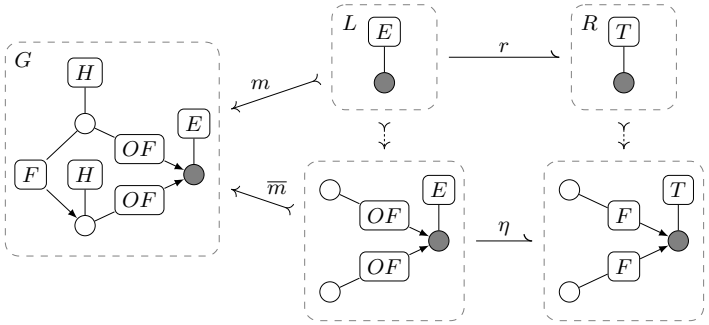


Fig. 2: A match of a universally quantified rule has to be extended until the entire neighbourhood of each quantified node is matched

In the following we formalize the notion of universally quantified rules as an extension of normal rules and introduce instantiations via a sequence of recursive instantiation steps.

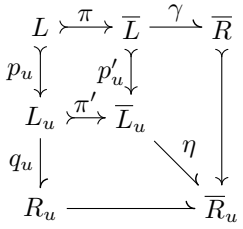
Definition 1 (Universally quantified rules). *A* universally quantified rule *is a pair $\rho = (r, U)$, where $r : L \rightharpoonup R$ is a partial morphism and U is a finite set of universal quantifications. A* universal quantification *is a pair $(p_u, q_u) = u \in U$ where $p_u : L \rightarrowtail L_u$ is a total injective morphism and $q_u : L_u \rightharpoonup R_u$ is a partial morphism satisfying the restriction that $q_u(p_u(x))$ is defined and has exactly one preimage in L_u for every $x \in L$.*

With $qn(u)$ we denote the set of quantified nodes *of u, which is the set of all $v \in V_L$ such that there is an edge incident to $p_u(v)$ which has no preimage in L. We denote the quantified nodes of a rule the same way, i.e. $qn(\rho) = \bigcup_{u \in U} qn(u)$. We require that $qn(u) \neq \emptyset$ for all $u \in U$.*

In the rest of the paper we will use UGTS to denote the extension of GTS with universally quantified rules.

Definition 2 (Instantiation of a universally quantified rule). *An instantiation of a universally quantified rule $\rho = (r, U)$ consists of a total injective morphism $\pi : L \rightarrowtail \overline{L}$ and a partial morphism $\gamma : \overline{L} \rightharpoonup \overline{R}$ and is recursively defined as follows:*

– *The pair $(id_L : L \rightarrowtail L, r)$, where id_L is the identity on L, is an instantiation of ρ.*
– *Let $(\pi : L \rightarrowtail \overline{L}, \gamma : \overline{L} \rightharpoonup \overline{R})$ be an instantiation of ρ and let $(p_u : L \rightarrowtail L_u, q_u : L_u \rightharpoonup R_u) = u \in U$. Furthermore, let $\overline{L}_u$ be the pushout of π, p_u and let $\overline{R}_u$ be the pushout of $\gamma \circ \pi$, $q_u \circ p_u$, as shown in the diagram to the right. Then $p'_u \circ \pi$ and the (unique) mediating morphism η are also an instantiation of ρ. We write $(p'_u \circ \pi, \eta) = (\pi, \gamma) \diamond u$ to indicate that the instantiation (π, γ) was extended by u.*



We say that the length of an instantiation is the number of steps performed to generate the instantiation, where (id_L, r) has a length of 0*.*

Example 2. Figure 3 shows a possible instantiation of the rule in Figure 1f. There is only one universal quantification u and this quantification is used once to generate the instantiation $(p'_u \circ id_L, \eta)$. Any further instantiation will add an additional node and OF-edge to $\overline{L}_u$ and an additional node and F-edge to $\overline{R}_u$. The universally quantified node (i.e. $qn(u)$) is marked grey. This means that η is only applicable if the grey node is matched to a node with degree (exactly) two. The rule application is performed by calculating the pushout of η (not r) and a valid matching m. The matching is only valid if all edges incident to the grey node have a preimage in $\overline{L}_u$, such that an application will always result in all incident OF-edges to be replaced by F-edges. Although the number of affected edges can be arbitrary large, the quantification it bounded to the neighbourhood of the grey node and therefore the change is still local.

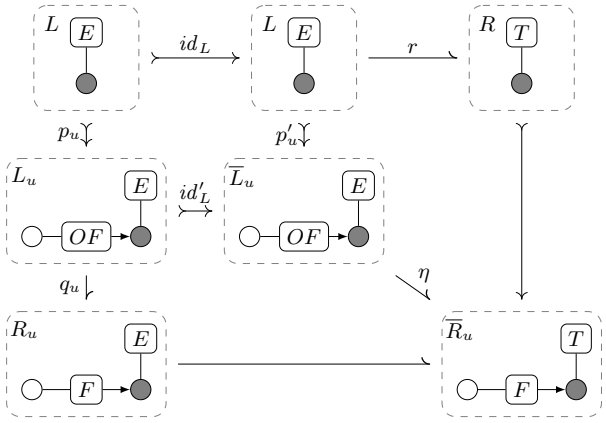


Fig. 3: A possible instantiation of the rule in Figure 1f

The order in which universal quantifications are used to generate instantiations can be neglected, since different sequences will still yield the same instantiation (up to

isomorphism). Therefore we can uniquely specify instantiations by the number each universal quantification in its sequence.

Definition 3 (Rule application). *Let ρ be a universally quantified rule. We say that ρ is applicable to a graph G, if there is an instantiation (π, γ) of ρ and a total injective match $m : \overline{L} \rightarrowtail G$, such that for every $x \in qn(\rho)$, there is no $e \in E_G$ incident to $m(\pi(x))$ without a preimage in $\overline{L}$. The application of ρ to G via m results in the graph H, the pushout of m and γ.*

We reuse the notation $G \Rightarrow G'$ to denote a rewriting step from G to G'. The previous definition introduces a restricted form of negative application condition since the existence of an edge, which cannot be mapped by a quantification, may block the application of a rule.

4 A Procedure for Coverability in UGTS

In this paper we focus our attention on verification problems that can be formulated as reachability and coverability decision problems. Given an initial configuration G_0 and a target configuration G_1 reachability consists in checking whether there exists a computation from G_0 to G_1. The coverability problem is similar to the reachability problem, but additionally relies on an ordering. In this paper we use the subgraph ordering, but there are other suitable orders such as the minor ordering or the induced subgraph ordering [24].

Definition 4 (Subgraph Ordering). *A graph G_1 is a subgraph of G_2, written $G_1 \subseteq G_2$, if there exists a partial, injective and surjective morphism from G_2 to G_1, written $\mu : G_2 \rightarrowtail G_1$. Such morphisms are called subgraph morphisms.*

Given a G, a subgraph can always be obtained by a sequence of node and edge deletions. Note that due to the morphism property every edge attached to a deleted node must be deleted as well. Using the subgraph ordering we can represent sets of configurations by minimal graphs and define two variants of the coverability problem.

Definition 5 (Upward Closure). *The* upward closure *of a set $\mathcal{S}$ of graphs is defined as $\uparrow\mathcal{S} = \{G' \mid G \subseteq G', G \in \mathcal{S}\}$. A set $\mathcal{S}$ is* upward-closed *if it satisfies $\mathcal{S} = \uparrow\mathcal{S}$. A* basis *of an upward-closed set $\mathcal{S}$ is a set $\mathcal{B}$ such that $\mathcal{S} = \uparrow\mathcal{B}$.*

Definition 6 (Coverability). *Let G_0, G_1 be two graphs. The* general coverability problem *is to decide whether from G_0 we can reach a graph G_2 such that $G_1 \subseteq G_2$.*

Let $\mathcal{G}$ a set of graphs and let $G_0, G_1 \in \mathcal{G}$. The restricted coverability problem *is to decide whether from G_0 we can reach a graph $G_2 \in \mathcal{G}$ such that $G_1 \subseteq G_2$ and every graph on the sequence from G_0 to G_2 is an element of $\mathcal{G}$.*

In other words, a configuration is coverable from some initial configuration if we can reach a configuration containing (as subgraph) a given pattern. Although general and restricted coverability are both undecidable, we can obtain decidability results by using a backward search introduced for well-structured transition systems [6,22] as

already shown in [7]. These systems rely on a *well-quasi-order (wqo)*, which is a transitive reflexive order $\leq$ such that there is no infinite, strictly decreasing sequence of elements and no infinite antichain, a sequence of pairwise incomparable elements, wrt. $\leq$. A direct consequence of this property is that every upward-closed set wrt. some wqo has a finite basis. It has been shown that the subgraph ordering is a well-quasi-order on $\mathcal{G}_k$, the class of graphs in which every undirected path has at most the length k [19]. We remark that the property does not hold if only directed paths are restricted.

The backward search presented in this paper is a version of the general backward search presented in [24] adapted to be compatible with UGTS. We denote the set of *predecessors* for a set of graphs $\mathcal{S}$ by $Pred(\mathcal{S}) = \{G' \mid \exists G \in \mathcal{S} \colon G' \Rightarrow G\}$. Furthermore we denote the predecessors reachable within multiple step by $Pred^*(\mathcal{S})$ and the *restricted predecessors* by $Pred_{\mathcal{G}}(\mathcal{S}) = Pred(\mathcal{S}) \cap \mathcal{G}$. We will present a procedure for UGTS to compute so-called effective pred-basis and effective $\mathcal{G}_k$-pred-basis. An *effective pred-basis* for a graph G is a finite basis $pb(G)$ of $\uparrow Pred(\uparrow\{G\})$ and an *effective $\mathcal{G}_k$-pred-basis* is a finite basis $pb_k(G)$ of $\uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$. Using the effective $\mathcal{G}_k$-pred-basis the backward search will terminate and compute a finite basis $\mathcal{B}$. If $G \in \uparrow\mathcal{B}$, then G covers a configuration of $\mathcal{S}$ in $\Rightarrow$ (general coverability). If $G \notin \uparrow\mathcal{B}$, then G does not cover a configuration of $\mathcal{S}$ in $\Rightarrow_{\mathcal{G}_k}$ (no restricted coverability), where $\Rightarrow_{\mathcal{G}_k}$ is the restriction $\Rightarrow \cap (\mathcal{G}_k \times \mathcal{G}_k)$. By using the effective pred-basis the backward search computes a finite basis for $Pred^*(\mathcal{S})$, but is not guaranteed to terminate.

The computation of a $\mathcal{G}_k$-pred-basis is performed by Procedure 1. We assume that for a graph G and a rule ρ there is an upper bound on the length of instantiations necessary to compute a backward step and write $bound_\rho(G)$ to denote such an upper bound. The existence of this upper bound is shown later on in Proposition 1. The result of a backward step is a finite set $\mathcal{S}$ of graphs such that $Pred(\uparrow\{G\}) \subseteq \uparrow\mathcal{S}$.

Procedure 1 (Backward Step).
Input: A rule ρ and a graph G.
Procedure:

1. First compute all instantiations $(\pi : L \rightarrowtail \overline{L}, \gamma : \overline{L} \rightharpoonup \overline{R})$ of ρ up to the length $bound_\rho(G)$.
2. For each γ compute all subgraph morphisms $\mu : \overline{R} \rightarrowtail R'$. Note that it is sufficient to take a representative R' for each of the finitely many isomorphism classes.
3. For each $\mu \circ \gamma$ compute all total injective morphisms $m' : R' \to G$ (co-matches of R' in G).
4. For each such morphism m' calculate all minimal pushout complements G', $m : \overline{L} \rightarrowtail G'$ of m' and $\mu \circ \gamma$ where m is injective and G' is an element of $\mathcal{G}_k$. Drop all G' where m does not satisfy the application condition of Definition 3, i.e. there is an edge incident to a quantified node which is not in the matching.

Result: The set of all graphs not dropped in Step 4, written $pb_k(G)$.

The motivation behind Step 2 is that G represents not just itself but also its upward closure. Therefore, the rule must also be applied to every graph larger than G. Instead of using partial co-matches we concatenate with subgraph morphisms to simulate this behaviour.

The procedure for a single backward step can be used to define a backward search procedure for the coverability problem for UGTS. The procedure exploits the property

that, even if compatibility is not satisfied, $Pred(\uparrow\mathcal{S}) \subseteq \uparrow Pred(\uparrow\mathcal{S})$ still holds for every set of graphs $\mathcal{S}$. We can iteratively compute backward steps for all minimal graphs G of $\uparrow\mathcal{S}$ and check that no initial state is reached backwards.

Procedure 2 (Backward Search).
Input: A natural number k, a set $\mathcal{R}$ of graph transformation rules and a finite set of final graphs $\mathcal{F}$. Start with the working set $\mathcal{W} = \mathcal{F}$.
Backward Step: For each $G \in \mathcal{W}$ add all graphs of $pb_k(G)$ to $\mathcal{W}$ and minimize $\mathcal{W}$ by removing all graphs H' for which there is a graph $H'' \in \mathcal{W}$ with $H' \neq H''$ and $H'' \subseteq H'$. Repeat this backward steps until the sequence of working sets $\mathcal{W}$ becomes stationary, i.e. for every $G \in \mathcal{W}$ the computation of the backward step using G results in no change of $\mathcal{W}$.
Result: The resulting set $\mathcal{W}$ contains minimal representatives of graphs from which a final state is coverable. This set may be an over-approximation, even without quantified rules.

To show the termination of Procedure 1 and 2 it is important to show the existence of a bounding function $bound_\rho()$. By the following proposition this function exists for every rule ρ, but as we will show later this bound can be tightened in most cases.

Proposition 1. *Let ι be an instantiation of length k of some rule ρ. If k is larger than the number of nodes and edges of G, then every graph computed by the backward application of ι is already represented by the backward application of an instantiation of lower length.*

The following two lemmas prove that Procedure 1 computes a finite basis of an over-approximation of the restricted predecessors.

Lemma 1. *The set $pb_k(G)$ is a finite subset of $Pred(\uparrow\{G\})$ and $pb_k(G) \subseteq \mathcal{G}_k$.*

Lemma 2. *It holds that $\uparrow pb_k(G) \supseteq \uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$.*

We recapitulate our main result in the following proposition.

Proposition 2. *For each graph G, $pb_k(G)$ is an effective $\mathcal{G}_k$-pred-basis. Furthermore, Procedure 2 terminates and computes an over-approximation of all configurations in $\mathcal{G}_k$ from which a final configuration is coverable.*

Proof. By Lemma 1 and 2 we know that $\uparrow pb_k(G) = \uparrow Pred_{\mathcal{G}_k}(\uparrow\{G\})$ and thus $pb_k(G)$ is a $\mathcal{G}_k$-pred-basis. According to Proposition 1 for every $\rho \in \mathcal{R}$ the number of necessary instantiation steps is bounded by $bound_\rho(G)$, thus, the number of instantiations is fine. For each instantiation the minimal pushout complements restricted to $\mathcal{G}_k$ are finite and computable. Since the subgraph ordering is decidable the minimization is computable and $pb_k(G)$ is effective.

Since the subgraph ordering is a wqo on $\mathcal{G}_k$, every infinite increasing sequence of upward-closed set becomes stationary. The upward-closures of the working sets $\mathcal{W}$ form such an infinite increasing sequence, thus the termination criteria of Procedure 2 will be satisfied at some point. □

A Variant of $pb_k()$ Without Path Bound

In Step 4 of Procedure 1 every graph which is not an element of $\mathcal{G}_k$ is dropped. This is needed to guarantee that the working set of Procedure 2 becomes stationary and the search terminates. However, this restriction can be dropped to obtain a backward search which solves the general coverability problem. Termination is not guaranteed, but correctness can be proven analogously to the restricted variant, as already shown in [24]. Let $pb()$ be Procedure 1 without the restriction to $\mathcal{G}_k$. We summarize the decidability of this second variant in the following proposition.

Proposition 3. *For each graph G, $pb(G)$ is an effective pred-basis. Furthermore, when using $pb()$ instead of $pb_k()$, Procedure 2 computes an over-approximation of all configurations from which a final configuration is coverable.*

Experimental Results

We added support for universally quantified rules to the UNCOVER tool. This tool can perform the backward search for the subgraph ordering and the minor ordering (a coarser order compared to subgraphs). Both variants of the backward search are implemented, but a timeout might occur when using the unresticted variant. However, given the rules in Figure 1 and the error graphs in Figure 4 the unrestricted variant terminates after 12 seconds and results in a set of 12 minimal graphs. Two of these graphs are the initial error graphs and two other computed graphs are shown in Figure 5. Every minimal graph contains a node in the state E. Since initially no philosopher is eating, the initial configuration is not represented and none of the initial error graphs is reachable. This proves that two adjacent philosophers cannot be eating at the same time.

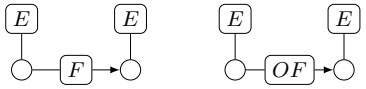


Fig. 4: Two error configurations in the Dining Philosophers Problem

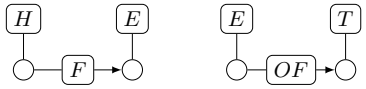


Fig. 5: Two other error graphs computed by the backward search

5 Optimizations

In this section we discuss and formalize some optimizations that can be applied to the basic backward procedure described in the previous section.

Lifting the Application Condition to a Post Conditions In Procedure 1 the application condition is checked in Step 4 for each pushout complement. However, by lifting the application condition over the instantiation we can check beforehand whether the backward step yields new graphs. We show the lifting in the following lemma.

Lemma 3. *Let ρ be a rule, $(\pi : L \rightarrowtail \overline{L}, \gamma : \overline{L} \rightharpoonup \overline{R})$ an instantiation of ρ and $m : \overline{R} \rightarrowtail G$ a co-match of the instantiation to some graph G. If there is a node $x \in qn(\rho)$ where $m(\gamma(\pi(x)))$ is defined and attached to an edge e without preimage in $\overline{R}$, then there is no pushout complement H of γ, m satisfying the condition of Definition 3.*

Tightening the Upper Bound of Instantiations The bound on the length of instantiations proven to exist in Proposition 1 can be improved depending on the rule used. Let $\rho = (r : L \rightharpoonup R, U)$ be a rule. Obviously $bound_\rho(G) = 0$ if $U = \emptyset$. The same holds if instantiations only increase the left side of the rule, i.e. for every $u \in U$ given the instantiation $(id_L, r) \diamond u = (\pi : L \rightarrowtail \overline{L}_u, \gamma : \overline{L}_u \rightharpoonup \overline{R}_u)$, the graphs $\overline{R}_u$ and R are isomorphic.

A more common situation is that quantifications do not add edges to the right side of the instantiations which are solely incident to nodes of the original rule r. This is case for all rules used in Example 1. The bound can be reduced as shown below.

Lemma 4. *Let $\rho = (r : L \rightharpoonup R, U)$ and let $(id_L, r) \diamond u = (\pi : L \rightarrowtail \overline{L}_u, \gamma : \overline{L}_u \rightharpoonup \overline{R}_u)$. If for every $u \in U$ every edge $e \in \overline{R}_u$ without preimage in R is connected to a node $v \in \overline{R}_u$ without preimage in R, then $bound_\rho(G) = |V_G|$.*

Optimization by Preparation The general framework in [24] uses a preparation step in the backward search to compute the concatenation of rules and subgraph morphisms performed in Step 2 of Procedure 1. This is not fully possible with universally quantified rules since the instantiations are generated within the backward steps. However, the preparation step can be performed for rules without universal quantifications. For rules with quantification the inner rule morphism can be concatenated with subgraph morphisms to partially prepare the rule. It can also be show that any concatenation of an instantiation and a subgraph morphism which is also a subgraph morphism, will not yield new graph in the backward step and thus can be dropped. This also holds for rules with universal quantification if all possible instantiations are also subgraph morphisms.

6 Conclusions and Related Work

In this paper we introduced a categorical formalization for an extension of graph transformation systems with universally quantified rules built on the single pushout approach. These rules are powerful enough to model distributed algorithms which use broadcast communication. A similar concept are adaptive star grammars [20] where the left-hand side of a rule is a star, i.e. a designated center node connected to a set of other nodes. Arbitrary large graphs can be matched by cloning parts of the star, which is – apart of the restriction to stars – one of the main differences to our approach. Technically our instantiations are a special form of amalgamated graph transformations [9], a technique to merge rules.

The backward search procedure presented in this paper is an extension of [24] with universally quantified rules and can be used for the verification of distributed algorithms, similar to [14]. There the induced subgraph ordering was used, which was also

shown to be compatible with the framework in [24]. However, our quantifications differ as we have a stronger negative application condition such that the induced subgraph ordering is not enough to cause our UGTS to satisfy the compatibility condition. This also causes the approached to differ in expressiveness. In general our approach should be compatible with the induced subgraph ordering and the minor ordering, but we did not yet investigated this.

Parameterized verification of combinations of automata- and graph-based models of distributed systems has been studied, e.g. in [10,4,15,16,13,12]. In [5] we applied graph-based transformations to model intermediate evaluations of non-atomic mutual exclusion protocols with universally quantified conditions. The conditions are not defined however in terms of graph rewrite rules. Semi-decision procedures can be defined by resorting to upward closed abstractions during backward search (monotonic abstraction as in [11]). In [10] we studied decidability of reachability and coverability for a graph-based specification used to model biological systems. Among other results, we proved undecidability for coverability for graph rewrite systems that can only increase the size of a configuration. Reachability problems for graph-based representations of protocols have also been considered in [4] where symbolic representations combining a special graph ordering and constraint-based representation of relations between local data of different nodes have been used to verify parameterized consistency protocols. Coverability for GTS is studied in [8] where it was proved that it is decidable for bounded path graphs ordered via subgraph inclusion. A model with topologies represented as acyclic directed graphs has been presented in [1]. Coverability for automata-based models of broadcast communication has recently been studied in [15,16,13,18,12]. In the context of program analysis approximated backward search working on graphs representing data structures with pointers have been considered in [2]. In this setting approximations are defined via edges or node deletion.

References

1. P. A. Abdulla, M. F. Atig, and O. Rezine. Verification of directed acyclic ad hoc networks. In *FMOODS/FORTE*, pages 193–208, 2013.
2. P. A. Abdulla, J. Cederberg, and T. Vojnar. Monotonic abstraction for programs with multiply-linked structures. *Int. J. Found. Comput. Sci.*, 24(2):187–210, 2013.
3. P. A. Abdulla, G. Delzanno, and A. Rezine. Approximated parameterized verification of infinite-state processes with global conditions. *Formal Methods in System Design*, 34(2):126–156, 2009.
4. P. A. Abdulla, G. Delzanno, and A. Rezine. Automatic verification of directory-based consistency protocols with graph constraints. *Int. J. Found. Comput. Sci.*, 22(4), 2011.
5. P. A. Abdulla, N. Ben Henda, G. Delzanno, and A. Rezine. Handling parameterized systems with non-atomic global conditions. In *VMCAI'08*, volume 4905 of *LNCS*, pages 22–36. Springer, 2008.
6. P. A. Abdulla, K. Čerāns, B. Jonsson, and Y. Tsay. General decidability theorems for infinite-state systems. In *Proc. of LICS '96*, pages 313–321. IEEE, 1996.
7. N. Bertrand, G. Delzanno, B. König, A. Sangnier, and J. Stückrath. On the decidability status of reachability and coverability in graph transformation systems. In *RTA'12*, volume 15 of *LIPIcs*, pages 101–116. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

8. N. Bertrand, G. Delzanno, B. König, A. Sangnier, and J. Stückrath. On the decidability status of reachability and coverability in graph transformation systems. In *RTA*, pages 101–116, 2012.
9. P. Boehm, H. Fonio, and A. Habel. Amalgamation of graph transformations: A synchronization mechanism. *Journal of Computer and System Sciences*, 34:377 – 408, 1987.
10. G. Delzanno, C. Di Giusto, M. Gabbrielli, C. Laneve, and G. Zavattaro. The *kappa*-lattice: Decidability boundaries for qualitative analysis in biological languages. In *CMSB*, pages 158–172, 2009.
11. G. Delzanno and A. Rezine. A lightweight regular model checking approach for parameterized systems. *STTT*, 14(2):207–222, 2012.
12. G. Delzanno, A. Sangnier, and R. Traverso. Parameterized verification of broadcast networks of register automata. In *RP'13*, pages 109–121, 2013.
13. G. Delzanno, A. Sangnier, R. Traverso, and G. Zavattaro. On the complexity of parameterized reachability in reconfigurable broadcast networks. In *FSTTCS'12*, volume 18 of *LIPIcs*, pages 289–300. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.
14. G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *Proc. CONCUR '10*, pages 313–327. Springer, 2010. LNCS 6269.
15. G. Delzanno, A. Sangnier, and G. Zavattaro. Parameterized verification of ad hoc networks. In *CONCUR'10*, volume 6269 of *LNCS*, pages 313–327. Springer, 2010.
16. G. Delzanno, A. Sangnier, and G. Zavattaro. On the power of cliques in the parameterized verification of ad hoc networks. In *FOSSACS'11*, volume 6604 of *LNCS*, pages 441–455. Springer, 2011.
17. G. Delzanno and J. Stückrath. Parameterized verification of graph transformation systems with whole neighbourhood operations, 2014. arXiv:1407.4394.
18. G. Delzanno and R. Traverso. Decidability and complexity results for verification of asynchronous broadcast networks. In *LATA*, pages 238–249, 2013.
19. G. Ding. Subgraphs and well-quasi-ordering. *Jornal of Graph Theory*, 16:489–502, November 1992.
20. F. Drewes, B. Hoffmann, D. Janssens, M. Minas, and N. V. Eetvelde. Adaptive star grammars. In *Proc. of ICGT '06 (International Conference on Graph Transformation)*, pages 77–91. Springer, 2006. LNCS 4178.
21. H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic approaches to graph transformation—part II: Single pushout approach and comparison with double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol.1: Foundations*, chapter 4. World Scientific, 1997.
22. A. Finkel and P. Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, April 2001.
23. M. Heumüller, S. Joshi, B. König, and J. Stückrath. Construction of pushout complements in the category of hypergraphs. In *Proc. of GCM '10 (Workshop on Graph Computation Models)*, 2010.
24. Barbara König and Jan Stückrath. A general framework for well-structured graph transformation systems. In P. Baldan and D. Gorla, editors, *Proc. of CONCUR 2014*, volume 8704 of *LNCS*, pages 467–481. Springer, 2014.
25. K. S. Namjoshi and R. J. Trefler. Uncovering symmetries in irregular process networks. In Roberto Giacobazzi, Josh Berdine, and Isabella Mastroeni, editors, *VMCAI*, volume 7737 of *Lecture Notes in Computer Science*, pages 496–514. Springer, 2013.