Decidability of Resilience for Well-structured Graph Transformation Systems
(Technical Report)

Okan Özkan

University of Oldenburg, Oldenburg, Germany
o.oezkan@informatik.uni-oldenburg.de

Abstract. Resilience is a concept of rising interest in computer science and software engineering. For systems in which correctness w.r.t. a safety condition is unachievable, fast recovery is demanded. We ask whether we can reach a safe state in a bounded number of steps whenever we reach a bad state. In a well-structured framework, we investigate problems of this kind where the bad and safety conditions are given as upward/downward-closed sets. We obtain decidability results for graph transformation systems by applying our results for subclasses of well-structured transition systems. Moreover, we identify sufficient criteria of graph transformation systems for the applicability of our results.

Keywords: Resilience · Graph transformation systems · Decidability · Well-structured transition systems

1 Introduction

Resilience is a broadly used concept in computer science and software engineering (e.g., [11]). In general engineering systems, *fast recovery* from a degraded state is often termed as resilience, see, e.g., [16]. In view of the latter interpretation of resilience, we investigate on the question whether a SAFE state can be reached in a bounded number of steps from any BAD state (where BAD is not necessarily the complement of SAFE). This concept is meaningful for systems in which violation of SAFE cannot be avoided. Our notion of resilience generalizes correctness (e.g., [2,10,14]) w.r.t. a safety condition.

For modeling systems, we use *graph transformation systems (GTSs)* in the *single pushout approach (SPO)*, as considered, e.g., in [7], which provide visual interpretability but yet also a precise formalism. In this perception, system states are captured by graphs and state changes by graph transformations. Our goal is to obtain decidability results for GTSs by considering their induced *transition systems*. A transition system consists of a set of states of any kind (not necessarily graphs) and a transition relation on the state set.

Usually, the state set (set of graphs) is infinite. To handle infinite state sets, we employ the concept of well-structuredness studied, e.g., in [1,9]. A *well-structured transition system (WSTS)* is, informally, a transition system equipped

with a *well-quasi-order* satisfying that larger states simulate smaller states (also called *compatibility* condition) and that certain predecessor sets can be effectively computed. In this well-structured setting, *ideal-based* sets (*upward-* or *downward-closed sets*) play an important role. They enjoy a number of suitable properties for verification such as finite representation (of upward-closed sets) and closure properties. For WSTSs, the *ideal reachability (coverability) problem* is decidable [1,9], which is an integrant of our results.

Well-structuredness of GTSs is investigated, e.g., in [12] for several well-quasi-orders. The well-quasi-order we use is the subgraph order which permits strong compatibility but comes with the restriction of path-length-boundedness on the graph class.

We show decidability for subclasses of GTSs of bounded path length. Each subclass exhibits additional requirements, i.e., effectiveness or unreliability properties. Additionally, we identify sufficient criteria of GTSs for the applicability of the results.

More precisely, we consider the *explicit* resilience problem where the bound on the number of steps for recovery is given and the *bounded* resilience problem which asks whether there exists such a bound. These problems are formulated for marked GTSs each of which consists of a GTS together with a graph class closed under rule application and an INITial subset of graphs. We ask: Starting from any graph in INIT, whenever we reach a BAD graph, can we reach a SAFE graph in $\leq k$ (in a bounded number of) steps?

To illustrate the idea of our resilience concept, we give an example.

Example (circular process protocol). Consider a ring of three processes P_0, P_1, P_2 each of which has an unordered collection (multiset) containing commands. Each command belongs to a process and is labeled accordingly as c_0, c_1, or c_2. The protocol is described below. A formalization as GTS can be found in Sec. 4.

- The process P_0 is *liberal*, i.e., it can *initiate* (generate and forward) a command c_0.
- Every process P_i can *forward* a command c_j, $i \neq j$, not belonging to itself.
- If a process P_i receives a command c_i, it is *enabled* and can
  1. *execute* its specific process action, or
  2. *clear* all commands in its collection and forward a command of the next process, or
  3. *leave* the process ring (if $i \neq 0$) and forward a command of the next process.

  Afterwards, the command c_i is deleted.
- Any command may get *lost* in any state.

The process action of P_0 is to forward two commands, c_1 and c_2. The process action of P_1 (P_2) is to forward a command c_2 (c_1). The topology of the process ring changes when a process leaves the ring. Processes P_1 and P_2 can leave the ring only if the other process has not left the ring before. In Fig. 1, the initial state where every process P_i has one command c_i in its collection and the three

possible topologies are shown. A process P_i is represented by an edge labeled with P_i. The collections are represented by white nodes which may have loops labeled with c_i corresponding to the contained commands.

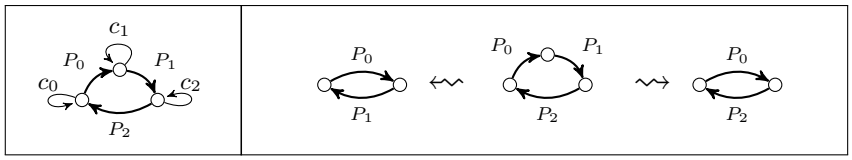


Fig. 1: Initial state and topologies of the circular process protocol.

Consider the following instances of the bounded resilience problem with the initial state as in Fig. 1:

BAD	¬AllEnabled	Command(c_2)	AllEnabled	NoCommand
SAFE	AllEnabled	Collection(c_0, c_1)	¬AllEnabled	No3Processes

For every instance of the bounded resilience problem, we are interested in a bound k for the number of steps needed for recovery. In the first instance, we ask whether we can reach a state where every process is enabled in $\leq k$ steps whenever we reach a state where this is not the case. In the second instance, we ask whether we can reach a state with a collection containing commands c_0, c_1 in $\leq k$ steps whenever we reach a state with a collection containing c_2. The third instance is the "dual" problem to the first one where the constraints for BAD and SAFE are exchanged. In the fourth instance, we ask whether we can reach a state containing no three processes in $\leq k$ steps whenever we reach a state without commands. One may ask:

- Does such a k exist? If so, what is the minimal k?
- Is there a generic method for problems of this kind?

We will answer these questions in Sec. 3 and 4.

This report is organized as follows: In Sec. 2, we recall preliminary concepts of GTSs and (WS)TSs. We show decidability of the resilience problems for subclasses of marked WSTSs in Sec. 3. In Sec. 4, we apply our results to marked GTSs. In Sec. 5, we give sufficient, rule-specific criteria for the applicability of our results. We present related concepts in Sec. 6 and close with a conclusion in Sec. 7. In the Appendix, a further example can be found.

2 Preliminaries

We recall the concepts used in this paper, namely graph transformation systems [7,6] and (in particular well-structured) transition systems [9].

2.1 Graph Transformation Systems

In the following, we recall the definitions of graphs, graph constraints, rules, and graph transformation systems [7,6].

A directed, labeled graph consists of a finite set of nodes and a finite set of edges where each edge is equipped with a source and a target node and where each node and edge is equipped with a label. Note that this kind of graphs are a special case of the hypergraphs considered in [12].

Definition 1 (graphs & morphisms). A *(directed, labeled) graph* (over a finite label alphabet $\Lambda = \Lambda_V \cup \Lambda_E$) is a tuple $G = \langle V_G, E_G, src_G, tgt_G, lab_G^V, lab_G^E \rangle$ with finite sets V_G and E_G of *nodes* (or *vertices*) and *edges*, functions $src_G, tgt_G : E_G \to V_G$ assigning *source* and *target* to each edge, and *labeling functions* $lab_G^V : V_G \to \Lambda_V$, $lab_G^E : E_G \to \Lambda_E$. A *(simple, undirected) path* in G of length ℓ is a sequence $\langle v_1, e_1, v_2 \ldots, v_\ell, e_\ell v_{\ell+1} \rangle$ of nodes and edges s.t. $src_G(e_i) = v_i$ and $tgt_G(e_i) = v_{i+1}$, or $tgt_G(e_i) = v_i$ and $src_G(e_i) = v_{i+1}$ for every $1 \leq i \leq \ell$, and all contained nodes and edges occur at most once. Given graphs G and H, a *(partial graph) morphism* $g : G \rightharpoonup H$ consists of partial functions $g_V : V_G \rightharpoonup V_H$ and $g_E : E_G \rightharpoonup E_H$ which preserve sources, targets, and labels, i.e., $g_V \circ src_G(e) = src_H \circ g_E(e)$, $g_V \circ tgt_G(e) = tgt_H \circ g_E(e)$, $lab_G^V(v) = lab_H^V \circ g_V(v)$, and $lab_G^E(e) = lab_H^E \circ g_E(e)$ on all nodes v and egdes e, for which $g_V(v), g_E(e)$ is defined. Furthermore, if a morphism is defined on an edge, it must be defined on both incident nodes. The morphism g is *total* (*injective*) if both g_V and g_E are total (injective). If g is total and injective, we also write $g : G \hookrightarrow H$. The composition of morphisms is defined componentwise. A pair $\langle G \to C, G' \to C \rangle$ of morphisms is *jointly surjective* if every item of C has a preimage in G or G'.

Convention. We draw graphs as usual. Labels are indicated by a symbol or a color. In (partial) morphisms, we equip the image of a node with the same index. Nodes on which the morphism is undefined have no index.

We consider a special case of graph constraints [15,10], which are non-nested and based on positive ($\exists G$)/ negative ($\neg\exists G$) constraints. For simplicity, we call them also positive (negative) constraints.

Definition 2 (positive & negative constraints). The class of *positive (negative) (graph) constraints* is the smallest class of expressions which contains $\exists G$ (negative: $\neg\exists G$) for every graph G and is closed under $\vee$ and $\wedge$. A graph G *satisfies* $\exists G'$ if there exists a total, injective morphism $G' \hookrightarrow G$. The semantics of the logical operators are as usual. We write $G \models c$ if G satisfies the positive/negative constraint c. For a positive/negative constraint c, we denote by $[\![c]\!]$ the set of all graphs G of the considered graph class with $G \models c$.

Using jointly surjective morphisms, every positive constraint can algorithmically be converted into an equivalent "$\vee$-normal form" (for the proof, see Appendix).

Fact 1 (from $\wedge$ to $\vee$). For every positive contraint c, we can effectively construct a positive constraint c' of the form $\bigvee_{1 \leq i \leq n} \exists G_i$ s.t. $[\![c]\!] = [\![c']\!]$ and there exists no total, injective morphism $G_i \hookrightarrow G_j$ for $i \neq j$.

We use the single pushout (SPO) approach [7] with injective matches for modeling graph transformations.

Definition 3 (graph transformation). A *(graph transformation) rule* $r = \langle L \rightharpoonup R \rangle$ is a partial morphism from a graph L to a graph R. A *graph transformation system (GTS)* is a finite set of rules. A *transformation* $G \Rightarrow H$ from a graph G to a graph H applying a rule r at a total, injective *match morphism* $g : L \hookrightarrow G$ is given by a *pushout* as shown in Fig. 2 (1) (for existence and construction of pushouts, see, e.g., [7]). We write $G \Rightarrow_r H$ to indicate the applied rule, and $G \Rightarrow_{\mathcal{R}} H$ if $G \Rightarrow_r H$ for a rule r in the rule set $\mathcal{R}$.

In Fig. 2 (2), an example for a transformation is shown.

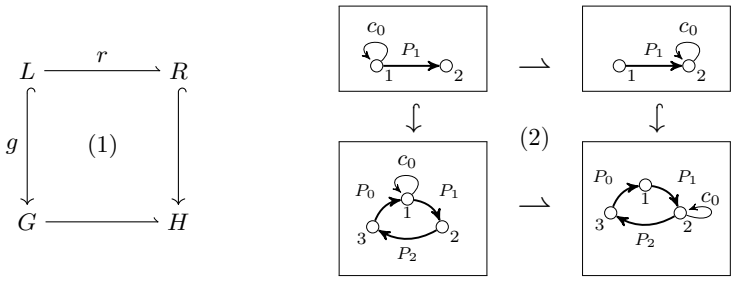


Fig. 2: Pushout scheme and example of a transformation.

2.2 Transition Systems and Well-structuredness

We recall the notion of transition systems.

Definition 4 (transition system). A *transition system (TS)* $\langle S, \to \rangle$ consists of a (possibly infinite) set S of *states* and a *transition relation* $\to \subseteq S \times S$. Let $\to^0 = \mathrm{Id}_S$ (identitiy on S), $\to^1 = \to$, and $\to^k = \to^{k-1} \circ \to$ for every $k \geq 2$. Let $\to^{\leq k} = \bigcup_{0 \leq j \leq k} \to^j$ for every $k \geq 0$. The *transitive closure* is given by $\to^* = \bigcup_{k \geq 0} \to^k$.

Often we are interested in the predecessors or successors of state set.

Definition 5 (pre- & postsets). Let $\langle S, \to \rangle$ be a transition system. For $S' \subseteq S$ and $k \geq 0$, we define $\mathrm{pre}^k(S') = \{s \in S \mid \exists s' \in S' : s \to^k s'\}$ and $\mathrm{post}^k(S') = \{s \in S \mid \exists s' \in S' : s' \to^k s\}$. Let $\mathrm{pre}^{\leq k}(S') = \bigcup_{j \leq k} \mathrm{pre}^j(S')$, $\mathrm{pre}^*(S') = \bigcup_{k \geq 0} \mathrm{pre}^k(S')$, $\mathrm{post}^{\leq k}(S') = \bigcup_{j \leq k} \mathrm{post}^j(S')$, and $\mathrm{post}^*(S') = \bigcup_{k \geq 0} \mathrm{post}^k(S')$. We abbreviate $\mathrm{post}^1(S')$ by $\mathrm{post}(S')$ and $\mathrm{pre}^1(S')$ by $\mathrm{pre}(S)$. A TS $\langle S, \to \rangle$ is *finite-branching* if $\mathrm{post}(s)$ is finite and computable for every given state s.

Several problems are undecidable for infinite-state TSs in general. However, interesting decidability results can be achieved if the system is well-structured [9,1,12]. A prerequisite for this concept is a well-quasi-order on the state set.

Definition 6 (well-quasi-order). A *quasi-order* is a reflexive, transitive relation. A *well-quasi-order (wqo)* over a set X is a quasi-order $\leq \subseteq X \times X$ s.t. every infinite sequence $\langle x_0, x_1, \ldots \rangle$ in X contains an increasing pair $x_i \leq x_j$ with $i < j$. A (well-)quasi-order is *decidable* if it can be decided whether $x \leq x'$ for all $x, x' \in X$.

In our setting, the subgraph order is of crucial importance.

Example 1 (subgraph order). The subgraph order $\leq$ is given by $G \leq H$ iff there is a total, injective morphism $G \hookrightarrow H$. Let S_ℓ be a graph class of bounded path length (with bound ℓ). The restriction of $\leq$ to S_ℓ is a wqo [12,4]. However, it is not a wqo on all graphs: The infinite sequence $\langle$  $\rangle$ of cyclic graphs of increasing length contains no increasing pair.

Assumption. From now on, we implicitly equip every set of graphs with the subgraph order. By "$\leq$" we mean either an abstract wqo or the subgraph order.

Upward- and downward-closed sets are of special interest.

Definition 7 (ideal & basis). Let X be a set and $\leq$ a quasi-order on X. For every subset A of X, we denote by $\uparrow A = \{x \in X \mid \exists a \in A : a \leq x\}$ the *upward-closure* and $\downarrow A = \{x \in X \mid \exists a \in A : x \leq a\}$ the *downward-closure* of A. An *ideal* $I \subseteq X$ is an upward-closed set, i.e., $\uparrow I = I$. An *anti-ideal* $J \subseteq X$ is a downward-closed set, i.e., $\downarrow J = J$. An (anti-)ideal is *decidable* if membership for every $x \in X$ is decidable. A *basis* of an ideal I is a subset $B \subseteq I$ s.t. (i) $\uparrow B = I$ and (ii) $b \neq b' \Rightarrow b \not\leq b'$ for all $b, b' \in B$.

Fact 2 (ideals of graphs). For every positive (negative) constraint c, $[\![c]\!]$ is an (anti-)ideal.

Ideals are, in general, infinite but can be represented by finite bases (a minimal generating set), similar to algebraic structures.

Fact 3 (finite basis [1, Lemma 3.3]). Every ideal has a basis and every basis is finite, provided that the superset is equipped with a wqo. Given a finite set A, a basis of $\uparrow A$ is computable, provided that the quasi-order is decidable.

Anti-ideals are the complements of ideals. Since an anti-ideal does not have an "upward-basis" in general, we will later demand that membership is decidable.

For well-structuredness, we demand that the wqo yields a simulation of smaller states by larger states. This condition is called compatibility.

Definition 8 (well-structured transition systems). Let $\langle S, \to \rangle$ be transition system and $\leq$ a decidable wqo on S. The tuple $\langle S, \leq, \to \rangle$ is a *well-structured transition system (WSTS)*, if:

(i) The wqo is *compatible* with the transition relation, i.e., for all $s_1, s_1', s_2 \in S$ with $s_1 \leq s_1'$ and $s_1 \to s_2$, there exists $s_2' \in S$ with $s_2 \leq s_2'$ and $s_1' \to^* s_2'$. If $s_1' \to^1 s_2'$, we say that it is *strongly compatible*. Both is illustrated in Fig. 3.
(ii) For every $s \in S$, a basis of $\uparrow \mathrm{pre}(\uparrow \{s\})$ is computable.

(a) Compatibility (b) Strong compatibility

Fig. 3: Visualization of (strong) compatibility.

A *strongly WSTS (SWSTS)* is a WSTS with strong compatibility.

Remark. For GTSs, strong compatibility is achieved by applying the same rule to the bigger graph. In contrast to the double pushout (DPO) approach [6], SPO has the suitable property that every rule is applicable to the bigger graph.

Assumption. Let $\langle S, \leq, \rightarrow \rangle$ be a well-structured transition system.

The set of ideals of S is closed under preset, union, and intersection.

Fact 4 (stability of ideals [1, Lemma 3.2]). For ideals $I, I' \subseteq S$, the sets $\mathrm{pre}^*(I)$, $I \cup I'$, and $I \cap I'$ are ideals. For SWSTSs, the sets $\mathrm{pre}(I)$, $\mathrm{pre}^{\leq k}(I)$ for every $k \geq 0$ are ideals.

An important point in our argumentation is the observation that every infinite, ascending sequence of ideals w.r.t. a wqo eventually becomes stationary.

Lemma 1 (Noetherian state set [1, Lemma 3.4]). For every infinite, ascending sequence $\langle I_0 \subseteq I_1 \subseteq \ldots \rangle$ of ideals, $\exists k_0 \geq 0$ s.t. $I_k = I_{k_0}$ for all $k \geq k_0$.

Abdulla et al. [1] exploit Lem. 1 to show the decidability of ideal reachability (coverability) for SWSTSs. The idea is to iteratively construct the sequence of the ideals $I^k = \mathrm{pre}^{\leq k}(I)$ until it becomes stable. This construction is carried out by representing ideals by bases. This argumentation is similarly feasible for WSTSs, see, e.g., [9, proof of Thm. 3.6].

Lemma 2 (ideal reachability [1, Thm. 4.1]). Given a basis of an ideal $I \subseteq S$ and a state s of a SWSTS, we can decide whether we can reach a state $s_I \in I$ from s. In particular, $\mathrm{pre}^{\leq k}(I) = \mathrm{pre}^*(I) \iff \mathrm{pre}^{\leq k+1}(I) = \mathrm{pre}^{\leq k}(I)$, and a basis of $\mathrm{pre}^*(I)$ is computable.

3 Decidability

We show the decidability of resilience problems for subclasses of SWSTSs by extending the idea in [13] to a systematic investigation.

In our setting, ideal-based sets of states play an important role.

Definition 9 (ideal-based). A set is *ideal-based* if it is (i) an ideal with a given basis, or (ii) a decidable anti-ideal. We denote by

(i) $\mathcal{I}$ the set of ideals with given bases, (ii) $\mathcal{J}$ the set of decidable anti-ideals.

We formulate resilience problems for marked WSTSs, i.e., WSTSs with a specified set INIT of inital states starting from which we investigate resilience.

Definition 10 (marked WSTS). A *marked WSTS* is a tuple $\langle S, \leq, \rightarrow, \mathrm{INIT}\rangle$ where $\langle S, \leq, \rightarrow\rangle$ is a WSTS and $\mathrm{INIT} \subseteq S$. If INIT is finite, we call it *fin-marked*.

Explicit Resilience Problem for WSTSs

Given: A marked WSTS $\langle S, \leq, \rightarrow, \mathrm{INIT}\rangle$, ideal-based sets SAFE, BAD $\subseteq S$, a natural number $k \geq 0$.

Question: $\forall s \in \mathrm{INIT} : \forall (s \rightarrow^* s' \in \mathrm{BAD}) : \exists (s' \rightarrow^{\leq k} s'' \in \mathrm{SAFE})$?

Bounded Resilience Problem for WSTSs

Given: A marked WSTS $\langle S, \leq, \rightarrow, \mathrm{INIT}\rangle$, ideal-based sets SAFE, BAD $\subseteq S$.

Question: $\exists k \geq 0 : \forall s \in \mathrm{INIT} : \forall (s \rightarrow^* s' \in \mathrm{BAD}) : \exists (s' \rightarrow^{\leq k} s'' \in \mathrm{SAFE})$?

For our further considerations, we regard requirements in order to obtain decidability, i.e., we consider the following subclasses of marked WSTSs.

Definition 11 (requirements). A marked WSTS $\langle S, \leq, \rightarrow, \mathrm{INIT}\rangle$ is

(1) *post*-effective* if INIT is finite and a basis of $\uparrow \mathrm{post}^*(\mathrm{INIT})$ is computable,
(2) *lossy* if $\downarrow \mathrm{post}^*(\mathrm{INIT}) = \mathrm{post}^*(\mathrm{INIT})$,
(3) $\bot$*-bounded (bottom-bounded)* if there is a natural number $\ell \geq 0$ s.t. $s_B \in \mathrm{post}^{\leq \ell}(s)$ for every $s \in S$ and every element s_B of a basis of S with $s \geq s_B$.

The requirement of post*-effectiveness describes the computability of the smallest reachable states from the initital states. The notion of lossiness means that the set of reachable states from the initital states is downward-closed. This is an abstraction from the lossiness concept in [9, p. 83]. Usually, the term "lossy" describes the circumstance that (almost) any piece of information of a state may get lost. Another kind of unreliability is $\bot$-boundedness which means that from every state, every smaller basis element (the bottom underneath) is reachable in a bounded number of steps. Thereby (almost) all information of a state may get lost in a bounded of number of steps.

The following lemma is crucial for many following proofs.

Lemma 3 (ideal-inclusion [13, Lemma 4]). Let $A \subseteq S$ be a set, $I \subseteq S$ an ideal, and $J \subseteq S$ an anti-ideal. Then, $A \cap J \subseteq I \iff \uparrow A \cap J \subseteq I$.

Applying this lemma to a basis B_I of an ideal I, we obtain that the inclusion $I \cap J \subseteq I'$ in an ideal I' can be checked by computing $B_I \cap J$ and then checking whether $B_I \cap J \subseteq I'$.

We give a characterization of post*-effectiveness via "anti-ideal reachability".

Proposition 1 (characterization of post*-effectiveness). For a class of finite-branching WSTSs, a basis of $\uparrow \mathrm{post}^*(s)$ is computable for every given state s iff the anti-ideal reachability problem is decidable, i.e., given a state s of a WSTS in the regarded class and a decidable anti-ideal J, it can be decided whether $\exists s' \in J : s \to^* s'$.

Proof. "$\Leftarrow$": Since the WSTS is finite-branching, for every $k \geq 0$, $\mathrm{post}^{\leq k}(s)$ is finite and computable. By Fact 3, for every $k \geq 0$, a basis of $\uparrow \mathrm{post}^{\leq k}(s)$ is computable. We compute the sequence of ideals $P_k = \uparrow \mathrm{post}^{\leq k}(s)$ until it becomes stationary (Lem. 1). The decidable stop condition (a condition which guarantees that we can terminate the algorithm) is to ask whether the decidable anti-ideal $S \setminus P_k$ is reachable from s. Namely, it holds: $\mathrm{post}^*(s) \cap S \setminus P_k = \varnothing \iff \mathrm{post}^*(s) \subseteq P_k \iff \uparrow \mathrm{post}^*(s) = P_k$.
"$\Rightarrow$": For deciding whether an anti-ideal J is reachable from s, we check whether $B_{\mathrm{post}}(s) \cap J = \varnothing$ where $B_{\mathrm{post}}(s)$ is a basis of $\uparrow \mathrm{post}^*(s)$. This is equivalent to $\mathrm{post}^*(s) \cap J = \varnothing$ by Lem. 3 as before. □

The characterization in Prop. 1 is used to show that Petri nets are post*-effective. It is well-known that Petri nets constitute SWSTSs [9, Thm. 6.1].

Example 2 (variations of Petri nets). (1) Petri nets (equipped with any finite set of initial states) are post*-effective by Prop. 1: Reachability for Petri nets is decidable and recursively equivalent to submarking reachability [8, p.6]. This corresponds to the anti-ideal reachability problem for Petri nets.
(2) *Lossy Petri nets* are Petri nets where in any state, one token may get lost at any place. Lossy Petri nets are lossy for every set of initial states.
(3) *Reset-lossy (mixed-lossy) Petri nets* are reset Petri nets [5] where in any state, all tokens (or one token) may get lost at any place. Reset-lossy (mixed-lossy) Petri nets are $\bot$-bounded (and lossy for every set of initial states).

For some results, we assume that a basis of the set of all states is given. This is only relevant if we use these basis elements for computations.

Notation. For a WSTS $\langle S, \leq, \to \rangle$ with a given basis of S, we write WSTS^B.

The next proposition shows how the requirements are related provided that a basis of the set of all states is given. The Venn diagram in Fig. 4 illustrates the relations of the subclasses corresponding to the requirements.

Proposition 2. Lossy ($\bot$-bounded) fin-marked WSTS^{B}s are post*-effective.

Proof. Let $\langle S, \leq, \to, \mathrm{INIT} \rangle$ be a lossy ($\bot$-bounded) fin-marked WSTS^B, B a basis of S, and $s \in \mathrm{INIT}$. In both cases, a basis of $\uparrow \mathrm{post}^*(s)$ is given by the set $B_{\mathrm{post}}(s) = \{s_B \in B : \uparrow \{s_B\} \text{ is reachable from } s\}$ which can be computed by Lem. 2. We show that $\uparrow B_{\mathrm{post}}(s) = \uparrow \mathrm{post}^*(s)$:
"$\supseteq$": Let s' be any element reachable from s. By definition of a basis, there exists a basis lement $s_B \leq s'$. It follows that $\uparrow \{s_B\}$ is reachable from s. We showed $\mathrm{post}^*(s) \subseteq \uparrow B_{\mathrm{post}}(s)$ and thus, $\uparrow \mathrm{post}^*(s) \subseteq \uparrow B_{\mathrm{post}}(s)$.

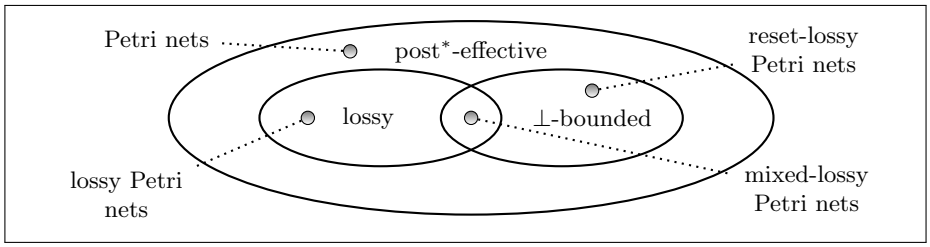


Fig. 4: Subclasses of fin-marked WSTS^{B}s.

"$\subseteq$": By definition of lossiness and $\bot$-boundedness, if $\uparrow\{s_B\}$ is reachable from s, also s_B is reachable from s. We showed $B_{\text{post}}(s) \subseteq \text{post}^*(s)$ and hence, $\uparrow B_{\text{post}}(s) \subseteq \uparrow \text{post}^*(s)$.

It holds

$$\begin{aligned}\uparrow\text{post}^*(\text{INIT}) &= \uparrow \bigcup_{s\in\text{INIT}} \text{post}^*(s) = \bigcup_{s\in\text{INIT}} \uparrow\text{post}^*(s)\\ &= \bigcup_{s\in\text{INIT}} \uparrow B_{\text{post}}(s) = \uparrow \bigcup_{s\in\text{INIT}} B_{\text{post}}(s).\end{aligned}$$

The set INIT is finite and, by Fact 3, $B_{\text{post}}(s)$ is finite for every $s \in S$. Hence, $\bigcup_{s\in\text{INIT}} B_{\text{post}}(s)$ is finite. By Fact 3, we can compute a basis of $\uparrow\text{post}^*(\text{INIT})$.□

Our main result for fin-marked SWSTSs terms sufficient criteria under which the resilience problems are decidable.

Theorem 1 (decidability for fin-marked SWSTSs).
Both resilience problems are decidable for fin-marked SWSTSs which are

(1) post*-effective if BAD $\in \mathcal{J}$, SAFE $\in \mathcal{I}$ (corresp. [13, Thm. 1]),
(2) lossy if BAD, SAFE $\in \mathcal{I}$.

The bounded resilience problem is decidable for fin-marked SWSTS^{B}s which are

(3) lossy and $\bot$-bounded if BAD $\in \mathcal{I}$, SAFE $\in \mathcal{J}$,
(4) $\bot$-bounded if BAD, SAFE $\in \mathcal{J}$.

Key Idea of the Proof. We compute a finite representation of post*(INIT) ∩ BAD for checking inclusion in a decidable ideal I which is a predecessor set of SAFE.

The proof structure is shown in Fig. 5: Lem. 4 states that for post*-effective (lossy) fin-marked SWSTSs, a finite representation of post*(INIT) ∩ BAD is computable, i.e., inclusion in a decidable ideal is decidable. In the case SAFE $\in \mathcal{I}$, the set $\text{pre}^{\leq k}$(SAFE) is a decidable ideal for every $k \geq 0$. (Lem. 5 shows the existence of bounds for the set of all predecessors of SAFE $\in \mathcal{J}$ provided that the SWSTS is $\bot$-bounded.) Prop. 3 shows that pre*(SAFE) constitutes a decidable ideal in the case SAFE $\in \mathcal{J}$ if the SWSTS^B is $\bot$-bounded.

The following lemma states that the inclusion of post*(INIT) ∩ BAD in an decidable ideal is decidable if we consider post*-effective in the case BAD $\in \mathcal{J}$ or lossy fin-marked WSTSs in the case BAD $\in \mathcal{I}$.

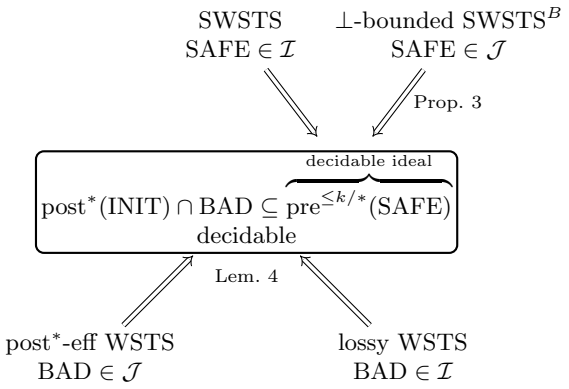


Fig. 5: Structure of the decidability proof for fin-marked SWSTSs.

Lemma 4 (checking inclusion). Let $\langle S, \leq, \rightarrow, \mathrm{INIT}\rangle$ be a fin-marked WSTS, $\mathrm{BAD} \subseteq S$, and $I \subseteq S$ be a decidable ideal. It is decidable whether $\mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq I$ provided that the fin-marked WSTS is (a) post*-effective and $\mathrm{BAD} \in \mathcal{J}$, (b) lossy and $\mathrm{BAD} \in \mathcal{I}$.

Proof. (a) We consider post*-effective fin-marked WSTSs and $\mathrm{BAD} \in \mathcal{J}$. It holds

$$\begin{aligned} & \mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq I & \\ \iff \quad & \uparrow \mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq I & \text{(Lem. 3)} \\ \iff \quad & B_{\mathrm{post}}(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq I & \text{(Lem. 3)} \end{aligned}$$

where $B_{\mathrm{post}}(\mathrm{INIT})$ is a basis of $\uparrow \mathrm{post}^*(\mathrm{INIT})$, i.e., $\uparrow B_{\mathrm{post}}(\mathrm{INIT}) = \uparrow \mathrm{post}^*(\mathrm{INIT})$. By post*-effectiveness, $B_{\mathrm{post}}(\mathrm{INIT})$ is computable. By Fact 3, $B_{\mathrm{post}}(\mathrm{INIT})$ is finite. The last inclusion is algorithmically checkable. We take out all elements of $B_{\mathrm{post}}(\mathrm{INIT})$ which are not in the decidable anti-ideal BAD and then check inclusion of the remaining elements in the decidable ideal I.

(b) We consider lossy fin-marked WSTSs and $\mathrm{BAD} \in \mathcal{I}$. We use the same idea as in the previous case, but we change the roles of $\mathrm{post}^*(\mathrm{INIT})$ and BAD. It holds

$$\begin{aligned} & \mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq I & \\ \iff \quad & \downarrow \mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq I & \text{(Def. lossy)} \\ \iff \quad & \downarrow \mathrm{post}^*(\mathrm{INIT}) \cap B \subseteq I & \text{(Lem. 3)} \end{aligned}$$

where B is a basis of BAD, i.e., $\uparrow B = \mathrm{BAD}$. We show that $\downarrow \mathrm{post}^*(\mathrm{INIT})$ is a decidable anti-ideal. For any $s \in S$, it holds $s \in \downarrow \mathrm{post}^*(\mathrm{INIT})$ iff $\mathrm{INIT} \cap \mathrm{pre}^*(\uparrow \{s\}) \neq \varnothing$. Since INIT is finite, we check whether $s' \in \mathrm{pre}^*(\uparrow \{s\})$ for every $s' \in \mathrm{INIT}$. The latter is decidable by Lem. 2. Hence, the last inclusion is decidable. We take out all elements of B which are not in $\downarrow \mathrm{post}^*(\mathrm{INIT})$ and then check inclusion of the remaining elements in the decidable ideal I. □

By the next lemma, $\bot$-boundedness implies that for any anti-ideal J, $\mathrm{pre}^*(J)$ is an ideal and $\mathrm{pre}^*(J) = \mathrm{pre}^{\leq k}(J)$ for a $k \geq 0$.

Lemma 5 (existence of bounds). For every $\bot$-bounded SWSTS and every anti-ideal J, there exists a $k \geq 0$ s.t. $\mathrm{pre}^*(J) = \uparrow \mathrm{pre}^*(J) = \mathrm{pre}^{\leq k}(J)$.

Proof (sketch). By Lem. 1, for every set A of states, there exists a $k_0 \geq 0$ s.t. $\uparrow \mathrm{pre}^*(A) = \uparrow \mathrm{pre}^{\leq k_0}(A)$. We show that there exists a constant $\ell \geq 0$ s.t. $\uparrow \mathrm{pre}^{\leq k}(J) \subseteq \mathrm{pre}^{\leq k+\ell}(J)$ for every anti-ideal J and every $k \geq 0$. Let $s' \geq s$ with $s \to^{\leq k} s_J \in J$, $k \geq 0$. By strong compatibility, there exists $\hat{s} \geq s_J$ with $s' \to^{\leq k} \hat{s}$. By definition of a basis, there exists a basis element $s_B \leq s_J$. Since $s_B \leq \hat{s}$ and by $\bot$-boundedness, we obtain $\hat{s} \to^{\leq \ell} s_B \in J$. By Lem. 1, there exists a $k_0 \geq 0$ s.t. $\uparrow \mathrm{pre}^*(J) = \uparrow \mathrm{pre}^{\leq k_0}(J)$. We obtain

$$\mathrm{pre}^*(J) \subseteq \uparrow \mathrm{pre}^*(J) = \uparrow \mathrm{pre}^{\leq k_0}(J) \subseteq \mathrm{pre}^{\leq k_0+\ell}(J) \subseteq \mathrm{pre}^*(J).$$

□

The following proposition identifies sufficient prerequisites s.t. $\mathrm{pre}^*(\mathrm{SAFE})$ constitutes a decidable ideal in the case $\mathrm{SAFE} \in \mathcal{J}$.

Proposition 3 (decidable ideals). For every $\bot$-bounded SWSTS^B and every decidable anti-ideal J, the set $\mathrm{pre}^*(J)$ is a decidable ideal.

Proof. By Lem. 5, $\uparrow \mathrm{pre}^*(J) = \mathrm{pre}^*(J)$. Thus, it is an ideal. Let $s \in S$. It holds

$$\begin{aligned}
& s \notin \mathrm{pre}^*(J) && \\
\iff\ & \nexists s' \in J : s \to^* s' && \text{(Def. preset)} \\
\iff\ & \mathrm{post}^*(s) \cap J = \varnothing && \text{(Def. postset)} \\
\iff\ & \uparrow \mathrm{post}^*(s) \cap J \subseteq \varnothing && \text{(Lem. 3)} \\
\iff\ & B_{\mathrm{post}}(s) \cap J \subseteq \varnothing && \text{(Lem. 3)}
\end{aligned}$$

where $B_{\mathrm{post}}(s)$ is a basis of $\uparrow \mathrm{post}^*(s)$. By Prop. 2, $\bot$-boundedness implies post^*-effectiveness w.r.t. any finite set of initial states provided that a basis of S is given. Hence, $B_{\mathrm{post}}(s)$ is computable. Thus, the last inclusion is decidable. □

We compile our preparatory results to prove Thm. 1.

Proof (of Thm. 1). Cases (1) & (2). By Fact 4, $\mathrm{pre}^{\leq k}(\mathrm{SAFE})$ is an ideal for every $k \geq 0$ since $\mathrm{SAFE} \in \mathcal{I}$. For every $k \geq 0$, $\mathrm{pre}^{\leq k+1}(\mathrm{SAFE}) = \mathrm{pre}(\mathrm{pre}^{\leq k}(\mathrm{SAFE})) \cup \mathrm{SAFE}$. By Def. 8 and Fact 3, a basis of $\mathrm{pre}^{\leq k}(\mathrm{SAFE})$ is iteratively computable. By Lem. 4, we can decide whether $\mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq \mathrm{pre}^{\leq k}(\mathrm{SAFE})$ for (1) post^*-effective fin-marked SWSTSs and (2) lossy fin-marked SWSTSs, respectively. By Lem. 1, the infinite ascending sequence $\mathrm{SAFE} \subseteq \mathrm{pre}^{\leq 1}(\mathrm{SAFE}) \subseteq \mathrm{pre}^{\leq 2}(\mathrm{SAFE}) \subseteq \ldots$ becomes stationary, i.e., there is a minimal $k_0 \geq 0$ s.t. $\mathrm{pre}^{\leq k_0}(\mathrm{SAFE}) = \mathrm{pre}^*(\mathrm{SAFE})$. By Lem. 2, we can also determine this k_0. Thus, we can determine the minimal number $k = k_{\min}$ s.t. $\mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq \mathrm{pre}^{\leq k}(\mathrm{SAFE})$ (if it exists) and also whether it exists. Hence, we can decide the

bounded resilience problem and given any k, we can check whether $k_{\min} \leq k$ to decide the explicit resilience problem.

Cases (3) & (4). By Lem. 5, for $\bot$-bounded SWSTSs, there exists a $k \geq 0$ s.t. $\mathrm{pre}^*(\mathrm{SAFE}) = \mathrm{pre}^{\leq k}(\mathrm{SAFE})$. Hence, checking bounded resilience is equivalent to testing inclusion in $\mathrm{pre}^*(\mathrm{SAFE})$. By Prop. 3, for $\bot$-bounded SWSTS^{B}s, $\mathrm{pre}^*(\mathrm{SAFE})$ is a decidable ideal since $\mathrm{SAFE} \in \mathcal{J}$. By Lem. 4, we obtain that checking $\mathrm{post}^*(\mathrm{INIT}) \cap \mathrm{BAD} \subseteq \mathrm{pre}^*(\mathrm{SAFE})$ is decidable for (3) lossy, $\bot$-bounded fin-marked SWSTS^{B}s and (4) post*-effective, $\bot$-bounded fin-marked SWSTS^{B}s, respectively. By Prop. 2, $\bot$-boundedness implies post*-effectiveness provided that a basis of the set of all states is given. □

4 Application to Graph Transformation Systems

We translate the results for WSTSs into the GTS setting.

The sets of positive and negative constraints are subsumed as ideal-based constraints.

Definition 12 (ideal-based constraints). We denote the set of positive (negative) constraints by $\mathcal{I}_c$ ($\mathcal{J}_c$). An *ideal-based constraint* is an element of $\mathcal{I}_c \cup \mathcal{J}_c$.

Recall that we consider the subgraph order as wqo. Path-length-boundedness on the graph class guarantees that the subgraph order yields a wqo.

Similarly to marked WSTSs, a marked GTS is a GTS together with a graph class closed under rule application and a subset INIT of graphs.

Definition 13 (marked GTS). A *marked GTS* is a tuple $\langle S, \mathcal{R}, \mathrm{INIT}\rangle$ where S is a (possibly infinite) set of graphs, $\mathcal{R}$ is a GTS with $\Rightarrow_{\mathcal{R}} \subseteq S \times S$, and $\mathrm{INIT} \subseteq S$. We speak of a marked GTS *of bounded path length*, shortly GTS$^{\mathrm{bp}}$, if S is of bounded path length and there exist $I \in \mathcal{I}$, $J \in \mathcal{J}$ (in the class of all graphs) s.t. $S = I \cap J$.

Remark. Ususally, one considers S as a decidable anti-ideal, as, e.g., in [12]. Then, the basis of S is given by the empty graph. By allowing $S = I \cap J$, we can consider more arbitrary bases of graphs. This is relevant for lossiness and $\bot$-boundedness. A basis of S is given by $B_I \cap J$ where B_I is basis of I.

Example 3 (starry sky). The rules $\langle \varnothing \rightharpoonup \textcircled{A} \rangle$ and $\langle {}_1\textcircled{A} \rightharpoonup {}_1\textcircled{A}\!\!\rightarrow\!\!\bullet \rangle$ together with the set of disjoint unions of unboundedly many star-shaped graphs (including isolated nodes) and any subset form a marked GTS$^{\mathrm{bp}}$.

We formulate the resilience problems for marked GTSs.

Explicit Resilience Problem for GTSs

Given: A marked GTS $\langle S, \mathcal{R}, \mathrm{INIT}\rangle$, ideal-based constraints Safe, Bad, a natural number $k \geq 0$.

Question: $\forall G \in \mathrm{INIT} : \forall (G \Rightarrow^* G' \models \mathrm{Bad}) : \exists (G' \Rightarrow^{\leq k} G'' \models \mathrm{Safe})$?

BOUNDED RESILIENCE PROBLEM FOR GTSS

Given: A marked GTS $\langle S, \mathcal{R}, \mathrm{INIT}\rangle$, ideal-based constraints $\mathrm{Safe}, \mathrm{Bad}$.
Question: $\exists k \geq 0 : \forall G \in \mathrm{INIT} : \forall (G \Rightarrow^* G' \models \mathrm{Bad}) : \exists (G' \Rightarrow^{\leq k} G'' \models \mathrm{Safe})$?

In the resilience problems for WSTSs, we considered ideal-based sets. We show that one can input ideal-based constraints instead.

Lemma 6 (ideal-based graph sets). Let $S = I \cap J$ be a graph class where $I \in \mathcal{I}$ and $J \in \mathcal{J}$. For every positive (negative) constraint c, $[\![c]\!] \in \mathcal{I}$ ($\mathcal{J}$).

Proof. By Fact 2, for every positive (negative) constraint c, the set $[\![c]\!]$ is an (anti-)ideal. Satisfaction ($\models$) of negative constraints is decidable. Let c be a positive constraint and $b = \bigvee_{G \in B} \exists G$ where B is a given basis of I. Then, $b \wedge c$ is a positive constraint. By Fact 1, we can compute a positive constraint c' s.t. $[\![c']\!] = [\![b \wedge c]\!]$ (in the class of all graphs) and c' is of the form $\bigvee_{1 \leq i \leq n} \exists G_i$ where $G_i \not\leq G_j$ for $i \neq j$. Since $J \in \mathcal{J}$, we can compute the set $\{G_i \in J : 1 \leq i \leq n\}$ which is a basis of $[\![c]\!]_S = \{G \in S : G \models c\}$. Hence, we can assume that a basis of $[\![c]\!]_S$ is given. □

Remark. More general constraints [15,10] do not constitute (anti-)ideals w.r.t. the subgraph order, in general. Consider, e.g., the "nested" constraint AllLoop = $\forall(\ 1\ \circlearrowleft, \exists(\ 1\ \circlearrowleft\!\!\leftarrow\ ))$ expressing that every node has a loop. The graph consisting of one node and one loop satisfies the latter constraint. However, the bigger graph consisting of two nodes and one loop does not satisfy it. (The smaller graph consisting of a single node does not satisfy it either.) Thus, $[\![\mathrm{AllLoop}]\!]$ is not an (anti-)ideal. Regarding the induced subgraph order [12], some "nested" constraints constitute ideals: The constraint $\exists(G, \bigwedge_{G^+ \in \mathrm{Ext}(G)} \neg\exists(G \hookrightarrow G^+))$ expresses that the graph G is an induced subgraph of the considered graph. Here $\mathrm{Ext}(G)$ is the set of all graphs G^+ obtained from G by adding one edge.

The following result of König & Stückrath terms a sufficient criterion for GTSs to be well-structured.

Lemma 7 (well-structured GTS [12, Prop. 7]). Every marked $\mathrm{GTS}^{\mathrm{bp}}$ induces a marked SWSTS^B (equipped with the subgraph order).

In particular, they give an effective procedure for obtaining a basis of $\mathrm{pre}(\uparrow\{G\})$ for every given graph G. Note that in [12], König & Stückrath consider labeled hypergraphs. However, the proof in our case is the same.

Convention. When speaking of a (fin-)marked $\mathrm{GTS}^{\mathrm{bp}}$, we consider the induced (fin-)marked SWSTS^B. We also adopt the terminology for "post*-effective", "lossy", and "$\bot$-bounded".

We apply our results from Sec. 3 to fin-marked $\mathrm{GTS}^{\mathrm{bp}}$.

Theorem 2 (decidability for fin-marked GTSs).
Both resilience problems are decidable for fin-marked $\mathrm{GTS}^{\mathrm{bp}}$s which are

(1) post*-effective if Bad $\in \mathcal{J}_c$, Safe $\in \mathcal{I}_c$,
(2) lossy if Bad, Safe $\in \mathcal{I}_c$.

The bounded resilience problem is decidable for fin-marked GTS^{bp}s which are

(3) lossy and $\bot$-bounded if Bad $\in \mathcal{I}_c$, Safe $\in \mathcal{J}_c$,
(4) $\bot$-bounded if Bad, Safe $\in \mathcal{J}_c$.

Proof. By Lem. 7 [12], every fin-marked GTS^{bp} induces a fin-marked SWSTS^B. Thus, the statements of Thm. 1 apply to GTS^{bp}s with the respective requirements. By Lem. 6, one can input ideal-based constraints instead of ideal-based sets. □

We illustrate our decidability results by an example.

Example: Circular Process Protocol

In Fig. 6, the formalization of the circular process protocol as GTS in Sec. 1 is shown. Note that each `Clear`-rule is undefined on the node which has a c_i-labeled loop. In a rule application, this node will be deleted and recreated. Note also that each `Leave`-rule identifies two nodes.

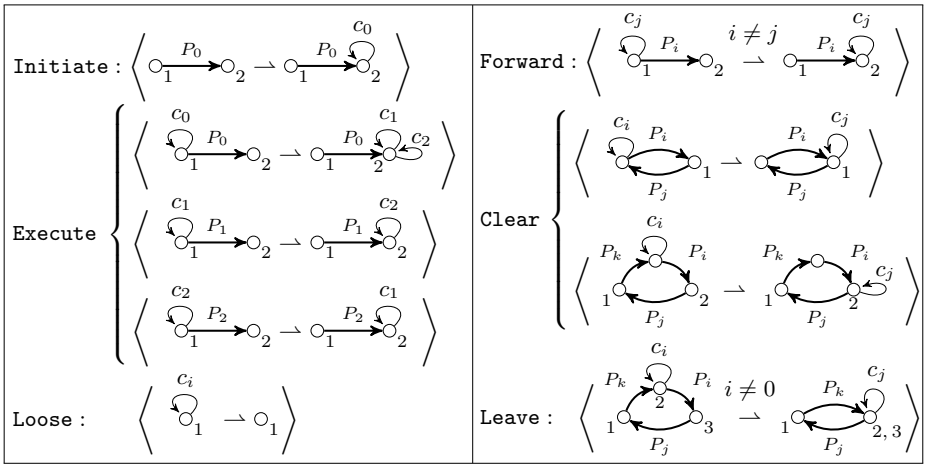


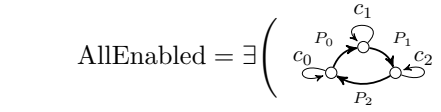
Fig. 6: Rules of the circular process protocol.

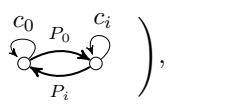
We consider all graphs with arbitrarily many commands of any kind (labeled with c_0, c_1, or c_2) in any collection, fitting to one of the topologies shown in Fig. 1. This graph class is of bounded path length. A basis of this graph class is given by the topologies without any commands as in Fig. 1. The marked GTS^{bp} is lossy since the rules for loosing a command c_i may be applied to any graph containing a command c_i. It is $\bot$-bounded since we can reach a graph with

the same topology but containing no commands by (i) initiating a command c_0, (ii) forwarding it to the collection of P_0, (iii) clearing all collections one after another, and (iv) loosing the only remaining command. By Prop. 2, it is post*-effective.

The example constraints for BAD and SAFE in Sec. 1 can be expressed as positive/negative constraints:

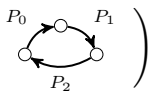
$$\text{AllEnabled} = \exists\Bigg(\quad \Bigg) \vee \bigvee_{i=1,2} \exists\Bigg(\quad \Bigg),$$





$$\text{Collection}(c_0, c_1) = \exists\Bigg(\quad \Bigg), \qquad \text{Command}(c_2) = \exists\Bigg(\quad \Bigg),$$

$$\text{No3Processes} = \neg\exists\Bigg(\quad \Bigg), \quad \text{NoCommand} = \bigwedge_{i=0,1,2} \neg\exists\Bigg(\quad \Bigg).$$



It can be verified that the given k's in the following table are minimal.

BAD	¬AllEnabled	Command(c_2)	AllEnabled	NoCommand
SAFE	AllEnabled	Collection(c_0, c_1)	¬AllEnabled	No3Processes
k	6	4	1	5

By clearing the collection of P_i, it is not enabled. Thus, for the third instance, $k_{\min} = 1$ since $k_{\min} \neq 0$. Using the algorithms presented in Sec. 3, we can compute $k_{\min}$ for the remaining cases.[1]

5 Rule-specific Criteria

We identify sufficient and handy GTS criteria for the requirements in Thm. 2. These criteria comprise properties of the rules.

Definition 14 (rule properties). A rule $\langle L \xleftarrow{p} R \rangle$ is *node-bijecitve* if p is bijective on the nodes. It is *preserving* if p is total and injective. A GTS is node-bijective (preserving) if all its rules are node-bijective (preserving).

For lossiness and $\bot$-boundedness, we consider sets of rules contained in a GTS in order to reach smaller graphs (but not smaller than basis elements).

Assumption. Let S be a graph class over Λ and B a basis of S.

Each lossy rule deletes one item (node or edge/loop) outside of a basis element.[2] Therefore they are constructed s.t. in each rule, a basis element is present.

[1] If SAFE $\in \mathcal{J} \setminus \mathcal{I}$, our method provides only the answer whether there is a bound k.

[2] In [12], "lossy rules" w.r.t. the minor order, i.e., edge contraction rules, are considered in order to obtain well-structuredness for GTSs.

Construction 1 (lossy rules). The set $\mathcal{R}_{\text{loss}}(S)$ of *lossy rules* w.r.t. S are constructed as follows.

(1) For graphs $G \in B$, $H \in \mathcal{H}_\Lambda$, the set $\mathcal{C}(G,H)$ is defined as all graphs C s.t. $\exists\langle G \hookrightarrow C, H \hookrightarrow C\rangle$ jointly surjective, and

$$\mathcal{H}_\Lambda = \left\{ \textcircled{x}, \quad \textcircled{x} \xrightarrow{a} \textcircled{y}, \quad \overset{a}{\circlearrowright}\textcircled{x} \;\middle|\; x,y \in \Lambda_V, a \in \Lambda_E \right\}.$$

(2) For every graph $C \in \mathcal{C}(G,H) \cap S$, every rule $\langle C \xrightarrow{p} p(C)\rangle$ where p is undefined on exactly one item (node or edge) which is not in (the image of) G and the identity otherwise, is a lossy rule.

A similar idea works for $\bot$-boundedness. Each bottom rule either deletes a node outside of a basis element, or deletes and recreates a node (with its incident edges) of a basis element.

Construction 2 (bottom rules). The set $\mathcal{R}_\bot(S)$ of *bottom rules* w.r.t. S are constructed as follows. For every basis element $G \in B$ and

(1) for every label $x \in \Lambda_V$ s.t. $G + \textcircled{x} \in S$, the rule $\langle G + \textcircled{x} \xrightarrow{p} G\rangle$ where p is undefined on the node $\textcircled{x}$ and the identity otherwise, is a bottom rule,[3]
(2) for every node $v \in V_G$, the rule $\langle G \xrightarrow{p} G\rangle$ where p is undefined on v and its incident edges, and the identity otherwise, is a bottom rule.

For $\bot$-boundedness, we additionally restrict the graph class. A graph class is *node-bounded* if the number of nodes in any graph of the class is bounded.

The following result shows that the rule-specific criteria are sufficient.

Theorem 3 (criteria). A marked GTS$^{\text{bp}}$ $\langle S, \mathcal{R}, \text{INIT}\rangle$ is

(1) post*-effective if (INIT is finite and) $\mathcal{R}$ is node-bijective or preserving,
(2) lossy if $\mathcal{R}_{\text{loss}}(S) \subseteq \mathcal{R}$,
(3) $\bot$-bounded if S is node-bounded and $\mathcal{R}_\bot(S) \subseteq \mathcal{R}$.

Proof. (1) If $\mathcal{R}$ is preserving, the statement follows by Fact 3. If $\mathcal{R}$ is node-faithful, the statement follows by the reduction in the proof of [3, Prop. 10]. For any graph G, a Petri net with initial marking is constructed s.t. reachability and the order of markings correspond to reachability via $\Rightarrow^*_{\mathcal{R}}$ from G and the subgraph order, respectively. Note that in [3], hypergraphs are considered. In our case, the places are given by $V_G \times V_G \times \Lambda_E$. The number of tokens in a place $\langle v, v', \lambda\rangle$ is given by

$$|\{e \in E : lab^E(e) = \lambda, src(e) = v, tgt(e) = v'\}|.$$

Each rule may give rise to a multiple number of transitions since a rule may have multiple matches. On the balance sheet, a transition takes tokens out of a place $\langle v, v', \lambda\rangle$ if the corresponding rule r deletes λ-labeled edges between v (source)

[3] The symbol "+" denotes the disjoint union of graphs.

and v' (target), and adds tokens in a place if r creates λ-labeled edges between v and v'. See [3] for further details.
Petri nets are post*-effective, see Ex. 2. Thus, a basis of $\uparrow \mathrm{post}^*(G)$ can be computed by computing the corresponding basis in the constructed Petri net.

(2) Let $G \geq G_B$ where $G \in S = I \cap J$, $G_B \in B$, and B is a basis of S (which is the intersection of a basis of I and J). We show that we can delete any item in $G \setminus i(G_B)$ where $i : G_B \hookrightarrow G$, i.e., for every item in $G \setminus i(G_B)$, there exists a lossy rule $r \in \mathcal{R}_{\mathrm{loss}}(S)$ which deletes this item (but preserves the rest excluding dangling edges) applied to G. For more clarity, we go through all cases. Note that for every graph G',

$$(*) \quad B \ni G_B \leq G' \leq G \in S \text{ implies } G' \in S$$

since $S = I \cap J$, J is downward-closed, and $\uparrow B \subseteq I$.
node deletion: Let v be the x-labeled node to be deleted. By (*), $G_B + \textcircled{x} \in S$. It holds $G_B + \textcircled{x} \in \mathcal{C}(G_B, \textcircled{x})$. Via the rule $\langle p : G_B + \textcircled{x} \rightharpoonup G_B \rangle \in \mathcal{R}_{\mathrm{loss}}(S)$ where p is undefined on $\textcircled{x}$ and the identity otherwise, we can delete v.
edge deletion: Let e be the a-labeled edge (but not a loop) to be deleted and $lab^V(src(e)) = x$ and $lab^V(tgt(e)) = y$. Let G^e_B be the smallest subgraph of G containing (the images of) G_B and e. The edge e can be incident to no node of G_B, only one node of G_B, or only nodes of G_B. In every case,

$$G^e_B \in \mathcal{C}(G_B, \textcircled{x} \xrightarrow{a} \textcircled{y}).$$

By (*), $G_B \leq G^e_B \leq G$ implies $G^e_B \in S$. Via the rule $\langle p : G^e_B \rightharpoonup p(G^e_B) \rangle \in \mathcal{R}_{\mathrm{loss}}(S)$ where p is undefined on e and the identity otherwise, we can delete e.
loop deletion: Let e be the a-labeled loop to be deleted and $lab^V(src(e)) = x = lab^V(tgt(e))$. Let G^e_B be the smallest subgraph of G containing (the images of) G_B and e. The loop e can be incident to a node of G_B or a node outside of G_B. In both cases,

$$G^e_B \in \mathcal{C}\left(G_B, \quad \begin{array}{c} a \\ \circlearrowleft \\ \textcircled{x} \end{array}\right).$$

By (*), $G_B \leq G^e_B \leq G$ implies $G^e_B \in S$. Via the rule $\langle p : G^e_B \rightharpoonup p(G^e_B) \rangle \in \mathcal{R}_{\mathrm{loss}}(S)$ where p is undefined on e and the identity otherwise, we can delete e.

Let $G \geq G' \in S$. There exists a sequence $G = G_0 \Rightarrow G_1 \Rightarrow \ldots \Rightarrow G_n = G'$ of node and edge deletions where $G_j \geq G_{j+1}$ for $0 \leq j \leq n-1$. By definition of a basis, there exists $G_B \in B$ s.t. $G_B \leq G'$. It follows that $G_B \leq G_j$ for $0 \leq j \leq n$ and the deleted items are outside of (the image of) G_B. Hence, the sequence of node and edge deletions from G to G' is also feasible via the lossy rules.

(3) Let $G \geq G_B$ where $G \in S = I \cap J$, $G_B \in B$, and B is a basis of S. Let $i : G_B \hookrightarrow G$. First we delete all nodes outside of $i(G_B)$. The graph class is compact. Hence, we can delete all nodes in $G \setminus i(G_B)$ in a bounded number of steps via the bottom rules $\langle p : G_B + \textcircled{x} \rightharpoonup G_B \rangle \in \mathcal{R}_{\perp}(S)$ where p is undefined on $\textcircled{x}$ and the identity otherwise (by (*), $G_B + \textcircled{x} \in S$). The remaining graph

consists only of (the image of) G_B and (unboundedly many) additional edges. By deleting every node in (the image of) G_B and recreating the node with its incident edges which are part of G_B, we get rid of the additional edges and reach (an isomorphic copy of) the basis element G_B. This is done by applying the bottom rules $\langle p : G_B \rightharpoonup G_B \rangle \in \mathcal{R}_\perp(S)$ where p is undefined on exactly one node of G_B and the identity otherwise. Roughly estimating, we can reach any basis element in $\leq n + \max_{G_B \in B} |V_{G_B}| \leq 2n$ steps where n is the bound on the number of nodes. □

Remark. A lossy/bottom rule intended for node deletion will delete dangling edges outside of (the image of) a basis element. A bottom rule of the "second type" is intended to delete dangling edges and restore items of the basis element.

Example 4 (criteria). (1) The GTS in Fig. 6 (circular process protocol) without the `Clear`- and `Leave`-rules is node-bijective. (2) The `Loose`-rules in Fig. 6 can be adapted (see Fig. 7) s.t. they fit in our definition of lossy rules, taking into account the three basis elements in Fig. 1.

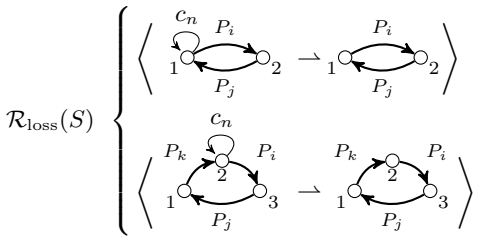


Fig. 7: The lossy rules of the circular process protocol.

(3) We adapt Ex. 3 (starry sky) s.t. the criterion for $\perp$-boundedness is fulfilled. Let D_n be the disjoint union of n A-labeled nodes and D_1^{loop} an A-labeled node with a single loop. We restrict the graph class to all graphs with exactly n A-labeled nodes and unboundedly many loops. The single basis element is the graph D_n. Consider the rule $\langle D_1 \hookrightarrow D_1^{\text{loop}} \rangle$ and the bottom rule $\langle D_n \xrightarrow{p} D_n \rangle$, i.e., deleting and recreating one node in D_n.

6 Related Concepts

The concept of resilience [16,11] is broadly used with varying definitions.

For modeling systems, we use SPO graph transformation as in [7].

Abdulla et al. [1] show the decidability of ideal reachability (coverability), eventuality properties and simulation in (labeled) SWSTSs. We use the presented algorithm as an essential integrant of our decidability proof.

Finkel & Schnoebelen [9] show that the concept of well-structuredness is ubiquitous in computer science by providing a large class of example models. They give several decidability results for well-structured systems with varying notions of compatibility, also generalizing the algorithm of [1] to WSTSs.

König & Stückrath [12] extensively study the well-structuredness of GTSs regarding three types of wqos (minor, subgraph, induced subgraph). All GTSs are strongly well-structured on graphs of bounded path length w.r.t. the subgraph order. This result enables us to apply our abstract results to GTSs. They regard Q-restricted WSTSs whose state sets have not to be a wqo but rather a subset Q of the states is a wqo. König & Stückrath develop a backwards algorithm based on [9] for Q-restricted WSTSs (GTSs).

Bertrand et al. [3] study the decidability of reachability and coverability for GTSs using, in parts, well-structuredness. A variety of rule-specific restrictions is investigated, e.g., containedness of node/edge-deletion rules. We use one of their results to obtain a sufficient criterion for post*-effectiveness. In contrast to [3], we stay in the framework of well-structured GTSs.

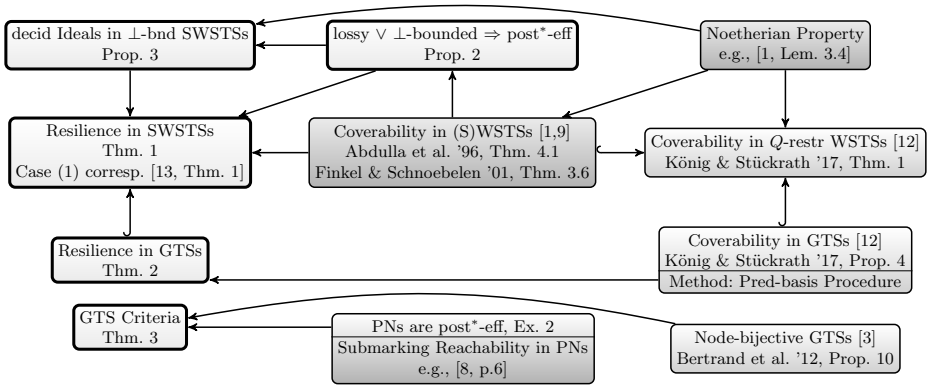


Fig. 8: Our results in the context of the theory of WSTSs and (ideal) reachability.

In Fig. 8, the main results of this paper (bold boxes) are placed in the context of known results. The arrows ($\longrightarrow$) mean "used for", the hooked arrows ($\hookrightarrow$) mean "instance of" or "generalized to". Our result for SWSTSs uses the well-known coverability algorithm [1,9] for (S)WSTSs which exploits the Noetherian property (a general concept for algebraic structures). For ⊥-bounded SWSTSs, we also employ the Noetherian property. On the level of GTSs, we use the predecessor-basis procedure of [12]. To the best of the author's knowledge, the considered notion of resilience was first studied in [13]. We extended the latter to a systematic investigation. The result for SWSTSs in [13] (Thm. 1) corresponds to case (1) of our Thm. 1. The result for GTSs in [13] (Thm. 2) is slightly less general than case (1) of our Thm. 2. For case (1) of Thm. 3, we use a result in [3] and well-known results for Petri nets [8].

7 Conclusion

We provided a systematic investigation on resilience problems obtaining decidability results for subclasses of marked GTSs by using the concept of well-structuredness. The used well-quasi-order on graphs is the subgraph order, i.e., a prerequisite is the path-length-boundedness on the graph class. The requirements for decidability are post*-effectiveness or a kind of unreliability (lossy, $\perp$-bounded). We identified sufficient rule-specific criteria for these requirements.

For future work, we will consider (1) possibilities of a modified approach for typed graphs [6], (2) other proof methods to handle nested constraints [10], and (3) other well-quasi-orders on graphs, e.g., the induced subgraph order [12].

Acknowledgment. I am grateful to Annegret Habel, Nick Würdemann, and the anonymous reviewers for their helpful comments.

References

1. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.K.: General decidability theorems for infinite-state systems. In: Proc. LICS 1996. pp. 313–321. IEEE (1996). https://doi.org/10.1109/LICS.1996.561359
2. Apt, K.R., Olderog, E.: Verification of Sequential and Concurrent Programs. Texts and Monographs in Computer Science, Springer (1991). https://doi.org/10.1007/978-1-4757-4376-0
3. Bertrand, N., Delzanno, G., König, B., Sangnier, A., Stückrath, J.: On the decidability status of reachability and coverability in graph transformation systems. In: 23rd Int. Conference on Rewriting Techniques and Applications (RTA'12). LIPIcs, vol. 15, pp. 101–116 (2012). https://doi.org/10.4230/LIPIcs.RTA.2012.101
4. Ding, G.: Subgraphs and well-quasi-ordering. J. Graph Theory **16**(5), 489–502 (1992). https://doi.org/10.1002/jgt.3190160509
5. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: ICALP 1998: Automata, Languages and Programming. LNCS, vol. 1443, pp. 103–115 (1998). https://doi.org/10.1007/BFb0055044
6. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science. An EATCS Series, Springer (2006). https://doi.org/10.1007/3-540-31188-2
7. Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., Corradini, A.: Algebraic approaches to graph transformation - part II: single pushout approach and comparison with double pushout approach. In: Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations, pp. 247–312. World Scientific (1997). https://doi.org/10.1142/9789812384720˙0004
8. Esparza, J., Nielsen, M.: Decidability issues for petri nets. BRICS Report Series **1**(8) (1994). https://doi.org/10.7146/brics.v1i8.21662
9. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. **256**(1-2), 63–92 (2001). https://doi.org/10.1016/S0304-3975(00)00102-X
10. Habel, A., Pennemann, K.: Correctness of high-level transformation systems relative to nested conditions. Math. Struct. Comput. Sci. **19**(2), 245–296 (2009). https://doi.org/10.1017/S0960129508007202

11. Jackson, S., Ferris, T.L.J.: Resilience principles for engineered systems. Syst. Eng. **16**, 152–164 (2013). https://doi.org/10.1002/sys.21228
12. König, B., Stückrath, J.: Well-structured graph transformation systems. Inf. Comput. **252**, 71–94 (2017). https://doi.org/10.1016/j.ic.2016.03.005
13. Özkan, O., Würdemann, N.: Resilience of well-structured graph transformation systems. In: Proceedings 12th Int. Workshop on Graph Computational Models. EPTCS, vol. 350, pp. 69–88 (2021). https://doi.org/10.4204/EPTCS.350.5
14. Poskitt, C.M., Plump, D.: Hoare-style verification of graph programs. Fundam. Informaticae **118**(1-2), 135–175 (2012). https://doi.org/10.3233/FI-2012-708
15. Rensink, A.: Representing first-order logic using graphs. In: Proc. ICGT 2004. LNCS, vol. 3256, pp. 319–335 (2004). https://doi.org/10.1007/978-3-540-30203-2_23
16. Trivedi, K.S., Kim, D.S., Ghosh, R.: Resilience in computer systems and networks. In: Proc. ICCAD 2009. pp. 74–77. IEEE/ACM (2009). https://doi.org/10.1145/1687399.1687415

Appendix

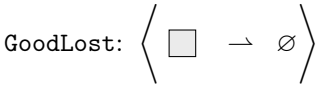
A Example: Logistic System

The circular process protocol in Sec. 4 satsifies all requirements. We give examples (variations of a logistic system) each of which is more "typical" for the respective requirement.

Consider a logistic system (LS1) with a unique *logistic center* C from where goods represented by □ are shipped to their *destination* D. The shipment can be direct or via an *interim storage* S. When an order is set, the good enters the system at the logistic center together with the planned route (edges labeled with *rou*) and the information of its destination and current location. This information is represented by edges labeled with *loc* and *dest*, respectively. Its destination D, a possible interim storage S, and routes may already be registered in the system or will enter when the order is set. The `SetOrder`-rules in Fig. 9 depict the process of setting an order with the planned route and information.

Some rules in Fig. 9 are presented in the Backus-Naur form, i.e., $\langle L \rightharpoonup R_1 | R_2 \rangle$ depicts the rules $\langle L \rightharpoonup R_1 \rangle$ and $\langle L \rightharpoonup R_2 \rangle$. For every dashed edge, one rule containing the edge in the left-handside and one rule not containing it is included, e.g., the last scheme depicts four rules. The goods can only be shipped along routes. A good can only be delivered to a destination if it fits to the destination information. The shipment is formalized by the `Ship`-rules shown in Fig. 10.

Some nodes are "C/S"-labeled meaning that both rules are included. This system is erroneous in the sense that a good can be shipped to an interim storage which is not the interim store on the planned route. Another "error" is the loss of a good which is depicted by the `GoodLost`-rule:



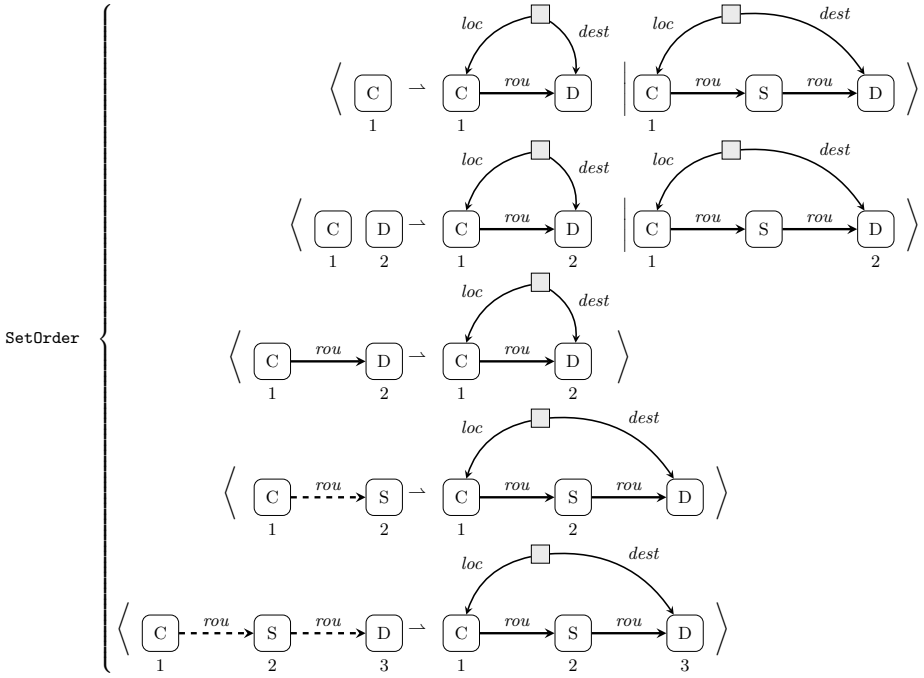


Fig. 9: The `SetOrder`-rules.

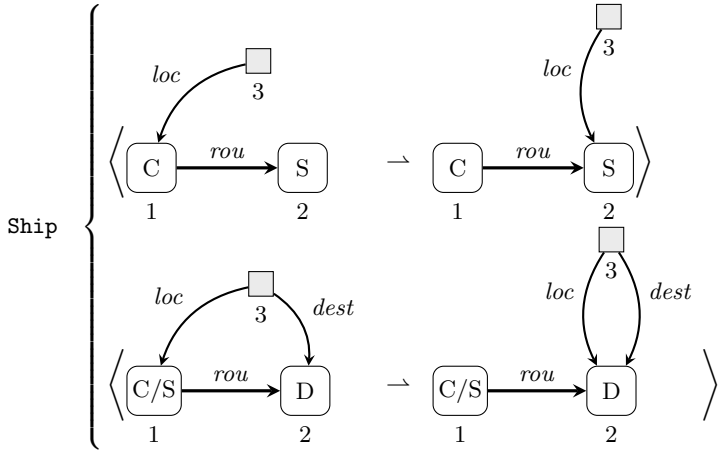


Fig. 10: The `Ship`-rules.

Let $\mathcal{R}_1 = \texttt{SetOrder} \cup \texttt{Ship} \cup \{\texttt{GoodLost}\}$. To obtain a marked GTS, we consider the intersection S of the downward-closure of all graphs reachable from the graph

C and all graphs bigger than the latter graph. The only basis element is the graph C. The graph class S is of path length ≤ 3. By definition, it is closed under application of rules in $\mathcal{R}$.

Observation 1. The GTS $\mathcal{R}_1$ together with the graph class S and any INIT $\subseteq S$ forms a marked GTS^{bp}.

We show that it is post*-effective for every finite set INIT.

Lemma 8. The fin-marked GTS^{bp} with the rule set $\mathcal{R}_1$ is post*-effective.

Proof. We show $\uparrow\text{post}^*(\text{INIT}) = \uparrow\text{post}^*_{\texttt{GoodLost}}(\text{INIT})$.
"$\supseteq$": It holds $\text{post}^*_{\texttt{GoodLost}}(\text{INIT}) \subseteq \text{post}^*(\text{INIT})$.
"$\subseteq$": Let $G \in \text{INIT}$, $G \Rightarrow^*_{\mathcal{R}} H$, and $G' \leq G$ s.t. G' contains no good (□). It holds $G' \in \text{post}^*_{\texttt{GoodLost}}(\text{INIT})$ and $H \geq G'$. Thus, $H \in \uparrow\text{post}^*_{\texttt{GoodLost}}(\text{INIT})$. Hence, $\text{post}^*(\text{INIT}) \subseteq \uparrow\text{post}^*_{\texttt{GoodLost}}(\text{INIT})$.
Since INIT is finite, the set $\text{post}^*_{\texttt{GoodLost}}(\text{INIT})$ is finite. By Fact 3, we can compute a basis of $\uparrow\text{post}^*_{\texttt{GoodLost}}(\text{INIT})$. Thus, the fin-marked GTS^{bp} is post*-effective. □

Variation LS2. In a more realistic scenario, there may occur "errors" in the logistic system, e.g., a good or information may be lost. The `Error`-rules depict the loss of (1) a good (`GoodLost`), the information about the (2) location or (3) destination of a good. Moreover, it can occur that (4) a route is inaccessible, (5) an interim storage is unusable, or (6) a destination is not detectable. The `Error`-rules also delete these defects from the system. Formally, the `Error`-rules are defined as in Fig. 11. Let $\mathcal{R}_3 = \texttt{SetOrder} \cup \texttt{Ship} \cup \texttt{Error}$. Using the `Error`-rules, we can delete any item except the logistic center C which is the only basis element. Thus, the marked GTS^{bp} is lossy for every set INIT.

Observation 2. The marked GTS^{bp} with the rule set $\mathcal{R}_3$ is lossy.

Remark. The lossy rules w.r.t. S are the adapted `Error`-rules in each of which the basis element C is present.

Variation LS3. We slightly adapt the model by representing a good as an □ -labeled edge pointing from its location to its destination. Thus the information about a good is compressed. We leave out the `Error`-rules except the last ones for deleting an S-labeled (D-labeled) node which we call `Inaccess`. The rules essentially stay the same. However, a `Ship`-rule has to be modified s.t. the destination D is present in the rule. Let $\mathcal{R}_4 = \texttt{SetOrder'} \cup \texttt{Ship'} \cup \texttt{Inaccess'}$ be the modified rule set.

Provided that the number of customers, i.e., destinations is "small", we can assume that number of D-labeled (and S-labeled) nodes is bounded. (This is the case if the logistic company has a small number of major customers.) In a realistic model, this does not apply to the number of goods which may be "very large" over the whole process.

In order to restrict the number of created nodes, we split the `SetOrder'`-rules into `SetOrder1`-rules which do not create nodes and `SetOrder2`-rules which

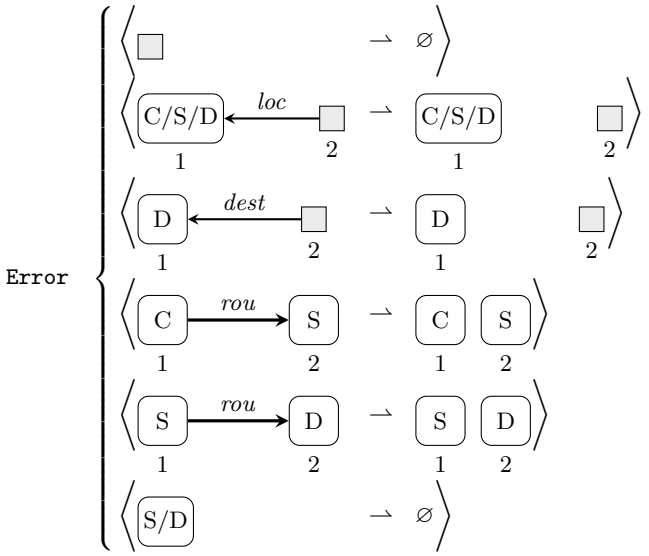


Fig. 11: The `Error`-rules.

create nodes.[4] For $n \geq 0$, let A_n be the rule-labeled automaton with states $q_0, \ldots, q_n$, transitions from q_i to q_{i+1} labeled with `SetOrder2`, and loops at q_i labeled with $\mathcal{R}_4 \setminus$ `SetOrder2`. In Fig. 12, the automaton A_n is exemplarily shown for $n = 2$. Let $\mathcal{R}_4^n$ denote the synchronization of $\mathcal{R}_4$ with A_n (see Sec.

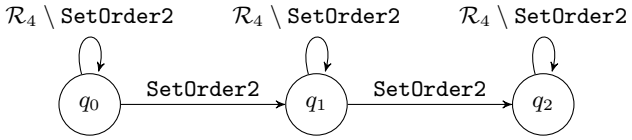


Fig. 12: The rule-labeled automaton A_n for $n = 2$.

3 in [13]). We have to consider graphs each of which contains exactly one q_i-labeled node for a number $1 \leq i \leq n$. Thus, a basis of this graph class is given by $B_n = \{\boxed{\text{C}}\,(q_i) : 1 \leq i \leq n\}$. Let S_n be the intersection of $\uparrow B_n$ with the downward-closure of all graphs reachable from the graph $\boxed{\text{C}}\,(q_0)$. Intuitively, the induced TS stays essentially the same as before with the restriction that the node-creating `SetOrder`-rules can be applied only boundedly often. Each

[4] It is possible to consider only `SetOrder1`-rules. However, this variation would have a "static" topology.

`SetOrder2`-rule creates ≤ 2 nodes. Thus, the number of nodes is bounded by $2n + 2$.

Observation 3. For every $n \geq 0$, the number of nodes in any graph in S_n is bounded.

The included `Inaccess`-rules ensure that any smaller basis element $\boxed{\text{C}}$ $\textcircled{q_i}$ is reachable in $\leq 2n$ transformation steps. Thus, the marked GTS^{bp} is $\bot$-bounded for every set $\text{INIT} \subseteq S_n$.

Observation 4. The marked GTS^{bp} with the rule set $\mathcal{R}_4^n$ and the graph class S_n is $\bot$-bounded.

B Proof

Proof (of Fact 1). By structural induction over positive constraints. Let c be a positive constraint. (i) For $c = \exists G$, the statement holds. (ii) For $c = c_1 \vee c_2$, let $c_1' = \bigvee_{1 \leq i \leq n_1} \exists G_i^1$ and $c_2' = \bigvee_{1 \leq i \leq n_2} \exists G_i^1$ be positive constraints s.t. $[\![c_1]\!] = [\![c_1']\!]$, $[\![c_2]\!] = [\![c_2']\!]$, and for $m = 1, 2$, $i \neq j$, there is no $G_i^m \hookrightarrow G_j^m$. Let B be the set containing all graphs G_i^m for $m = 1, 2$ and $1 \leq i \leq n_m$. The set B' is obtained from B by sorting out all graphs $G_i^m \in B$ s.t. there exists $G_j^k \hookrightarrow G_i^m$ for a number $1 \leq j \leq n_k$ where $m \neq k = 1, 2$. Let $c' = \bigvee_{G \in B'} \exists G$. It holds $[\![c]\!] = [\![c']\!]$. (iii) For $c = c_1 \wedge c_2$, let $c_1' = \bigvee_{1 \leq i \leq n_1} \exists G_i^1$ and $c_2' = \bigvee_{1 \leq i \leq n_2} \exists G_i^1$ be positive constraints s.t. $[\![c_1]\!] = [\![c_1']\!]$, $[\![c_2]\!] = [\![c_2']\!]$, and for $m = 1, 2$, $i \neq j$, there is no $G_i^m \hookrightarrow G_j^m$. For $m = 1, 2$, let B_m be the set containing all graphs G_i^m for $1 \leq i \leq n_m$. For every graph H holds:

$$
\begin{aligned}
& & H &\models c_1 \wedge c_2 & & \\
&\iff & \exists (G^1 \hookrightarrow H),\ & (G^2 \hookrightarrow H) \text{ where } G^1 \in B_1, G^2 \in B_2 & &\text{(Def. } \models) \\
&\iff & \exists (C \hookrightarrow H) :\ & \exists \langle G^1 \hookrightarrow C, G^2 \hookrightarrow C \rangle \text{ jointly} & &\text{(js)} \\
& & & \text{surjective where } G^1 \in B_1, G^2 \in B_2 & & \\
&\iff & H &\models \underbrace{\bigvee_{\langle G^1, G^2 \rangle \in B_1 \times B_2} \ \bigvee_{C \in \mathcal{C}(G^1, G^2)} \exists C}_{=: c'} & &\text{(Def. } \models)
\end{aligned}
$$

where $\mathcal{C}(G^1, G^2)$ is the set of all graphs C s.t. $\exists \langle G \hookrightarrow C,\ G' \hookrightarrow C \rangle$ jointly surjective. We show that (js) holds:
"$\Leftarrow$": Let $h : C \hookrightarrow H$ be an injective morphism and $\langle p_1 : G^1 \hookrightarrow C, p_2 : G^2 \hookrightarrow C \rangle$ jointly surjective. The morphisms $h \circ p_1 : G^1 \hookrightarrow H$, $h \circ p_2 : G^2 \hookrightarrow H$ are injective.
"$\Rightarrow$": Let $p_1 : G \hookrightarrow H$, $p_2 : G^2 \hookrightarrow H$ be injective morphisms. Let C be the graph $p_1(G^1) \cup p_2(G^2)$. By definition, the morphisms $p_1 : G^1 \hookrightarrow C$, $p_2 : G^2 \hookrightarrow C$ are injective and together jointly surjective. Moreover, there exists an injective morphism $C \hookrightarrow H$.
Thus, $[\![c]\!] = [\![c']\!]$. We show that for all graphs G^1, G^2, the set $\mathcal{C}(G^1, G^2)$ of graphs is finite and computable. By joint surjectivity, the number of nodes (edges) of any graph in $\mathcal{C}(G^1, G^2)$ is $\leq |V_{G^1}| + |V_{G^2}|$ (edges: $\leq |E_{G^1}| + |E_{G^2}|$). We can test all such graphs for joint surjectivity in the following way: For two graphs $G^1 \in B_1$, $G^2 \in B_2$, and a graph C to be tested, we compute all injective morphisms $G^1 \hookrightarrow C$ and $G^2 \hookrightarrow C$. We check whether every item of C has a preimage in G^1 or G^2. The set $B_1 \times B_2$ is finite. Hence, we can effectively construct the latter positive constraint c'. For minimality, we can sort out superfluos graphs as in (ii). □