Resilience of Well-structured Graph Transformation Systems*

Okan Özkan Nick Würdemann
Department of Computing Science
University of Oldenburg
Oldenburg, Germany
{o.oezkan,wuerdemann}@informatik.uni-oldenburg.de

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 investigate resilience problems of graph transformation systems. Our main contribution is the decidability of two resilience problems for well-structured graph transformation systems (with strong compatibility). We prove our results in the abstract framework of well-structured transition systems and apply them to graph transformation systems, incorporating also the concept of adverse conditions.

1 Introduction

Resilience is a broadly used concept in computer science and software engineering (e.g., [20]), and a basic concept for, e.g., industrial control systems [16] and mobile cyber-physical systems [13]. For systems in which *correctness* w.r.t. a safety condition SAFE is unachievable, *fast recovery* is demanded. We interprete fast recovery as reachability of the safety condition in a bounded amount of time steps. The intuitive approach is to start from any *error state*, i.e., a state in which $\neg$SAFE($\equiv$ ERROR) holds, and try to reach a state in which SAFE holds again as fast as possible.

Another approach to formalizing resilience is to ask whether the system can *withstand an adverse effect* rather than to ask whether fast recovery is possible from any error state. To formally capture adverse effects we consider an environment interacting with the system. In this setting, we investigate on the question whether a state satisfying SAFE can be reached in bounded time, starting from any state satisfying ENV, i.e., any state directly resulting from an environment interference.

For modeling systems we use *graph transformation systems (GTSs)*, as considered, e.g., in [6], which are a visual yet precise formalism. In this perception, system states are captured by graphs and state changes by graph transformations. Usually, the state set (the set of graphs reachable from a start graph) is infinite. To handle infinite state sets, we incorporate the concept of well-structuredness [1, 8, 10]. A *well-structured transition system (WSTS)* is informally a transition system equipped with a well-quasi-order (wqo) satisfying that larger states simulate smaller states. This allows us to abstract from both of the approaches towards resilience described above. In the setting of WSTSs, we define resilience problems for a given downward-closed set J (a BAD condition, e.g., ERROR or ENV) and an upward-closed set I (e.g., a safety property SAFE). Given an initial state s and a natural number k, the *explicit resilience problem* asks whether we can, starting from s, reach I in at most k steps whenever we reach J. The *bounded resilience problem* asks whether there exists a k such that *k-step resilience* is satisfied.

We show that both resilience problems (given a basis of the upward-closure of the reachable states) are decidable for *strongly* well-structured transition systems (SWSTSs). We propose an algorithm which computes the minimal k s.t. we can recover from any BAD state in at most k steps, or returns `false` if

*Supported by the German Research Foundation (DFG) through the Research Training Group (DFG GRK 1765) SCARE

B. Hoffmann and M. Minas (Eds.): Twelfth International
Workshop on Graph Computation Models (GCM 2021)
EPTCS 350, 2021, pp. 69–88, doi:10.4204/EPTCS.350.5

there exists no such k. It is based on the ideal reachability algorithm proposed by Abdulla et al. [1], and solves both resilience problems at the same time.

When applying these results to GTSs, we assume that the corresponding graph class is of bounded path length in order to obtain a SWSTS. This sufficient condition for a GTS to be strongly well-structured is shown by König & Stückrath in [10]. The wqo on graphs used in this case is the subgraph order, so $I = I_{\text{SAFE}}$ corresponds to a constraint stating existence of subgraphs. We incorporate adverse conditions by distinguishing system and environment rules, and considering $J = J_{\text{ENV}}$, the set of graphs directly resulting from the application of an environment rule.

The rest of this paper is organized as follows: We recall preliminary concepts in Sec. 2. In Sec. 3, we present the concept of resilience in the context of adverse conditions and identify abstract resilience problems. In Sec. 4, we prove decidability of resilience for strongly well-structured transition systems. We apply these results to graph transformation systems incorporating adverse conditions in Sec. 5. In Sec. 6, we present related work. We close with a conclusion and an outlook in Sec. 7.

2 Preliminaries

We recall the concepts used in this paper, namely *graph transformation systems* [6, 5] and (in particular *well-structured*) *transition systems* [8].

2.1 Graph Transformation Systems

In the following, we recall the definitions of graphs, graph conditions, rules, and graph transformation systems [6, 5]. A directed, labeled graph consists of a set of nodes and a 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 [10].

Definition 1 (graphs & graph morphisms)**.** A *(directed, labeled) graph* (over a finite label alphabet Λ) is a tuple $G = \langle V_G, E_G, src_G, tgt_G, lab^V_G, lab^E_G \rangle$, 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* $lab^V_G : V_G \to \Lambda$, $lab^E_G : E_G \to \Lambda$. A *(simple, undirected) path* p 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. Let $\ell(G)$ denote the length of a longest path in G. 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., $g_V \circ src_G(e) = src_H \circ g_E(e)$, $g_V \circ tgt_G(e) = tgt_H \circ g_E(e)$, $lab^V_G(v) = lab^V_H \circ g_V(v)$, and $lab^E_G(e) = lab^E_H \circ g_E(e)$ on all egdes e and nodes v, for which $g_E(e), lab^E(e), lab^V(v)$ is defined. Furthermore, if a morphism is defined on an edge, it must be defined on all 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.

We consider graph constraints [15, 9] whose validities are inherited to bigger/smaller graphs.

Definition 2 (positive & negative basic graph constraints)**.** The class of *positive (basic graph) constraints* is defined inductively: (i) $\exists G$ is a positive constraint where G is a graph, (ii) for positive constraints c, c', also $c \vee c'$, $c \wedge c'$ are positive constraints. Analogously, the *negative (basic graph) constraints* are defined by: (i) $\neg\exists G$ is a negative constraint for any graph G, (ii) for negative constraints c, c', also $c \vee c'$, $c \wedge c'$ are negative constraints. A graph G *satisfies* $\exists G'$ if there exists an 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.

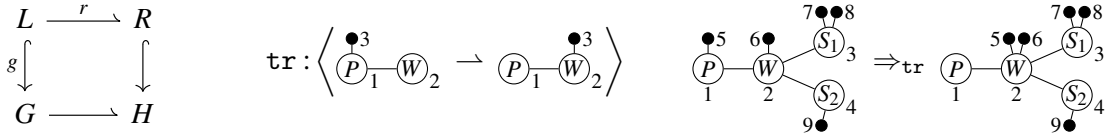
Remark. If c is a positive constraint, $\neg c$ is equivalent to a negative constraint, and vice versa.

Fact 1 (upward & downward inheritance)**.** *Let $G \hookrightarrow H$ be a total injective morphism, c be a positive constraint, and c' a negative constraint. If $G \models c$, then also $H \models c$. If $H \models c'$, then also $G \models c'$.*

We use the *single pushout (SPO)* approach [6, 10] with injective matches for modeling graph transformations. The reason for choosing SPO and not, e.g., the *double pushout approach (DPO)* [5] is that the dangling condition disturbs the compatibility condition of WSTS in Def. 10.

Definition 3 (rules & transformations)**.** A *(graph transformation) rule* $r = \langle L \rightharpoonup R \rangle$ (over a finite label alphabet Λ) is a partial morphism from L to R (both graphs over Λ). A *(direct) transformation* $G \Rightarrow H$ from a graph G to a graph H applying rule r at a total injective *match morphism* $g : L \hookrightarrow G$ is given by a *pushout* as shown in Fig. 1a (for existence and construction of pushouts, see, e.g., [6]). We write $G \Rightarrow_r H$ to indicate the applied rule, and $G \Rightarrow_{\mathscr{R}} H$ if $G \Rightarrow_r$ for a rule r contained in the rule set $\mathscr{R}$.

Note that we do not have any application conditions. The pushout of a rule application is visualized in Fig. 1a. An example for a rule is presented in Fig. 1b, and an application of that rule in Fig. 1c.



(a) Pushout for the transformation $G \Rightarrow_r H$

(b) A graph transformation rule `tr`

(c) An application of the rule `tr`

Figure 1: Pushout and example of a direct graph transformation.

GTSs are simply finite sets of rules. We specify the state set later.

Definition 4 (graph transformation system)**.** A *graph transformation system (GTS)* is a finite set of graph transformation rules.

2.2 Transition Systems

We recall the notion of transition systems. In Sec. 4, we prove our results on the level of transition systems and explicate the concept for graph transformation systems in Sec. 5.

Definition 5 (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 $\to^* = \bigcup_{k \geq 0} \to^k$.

The following definition shows how any GTS can be interpreted as a TS.

Definition 6 (graph transition system)**.** Let $\mathscr{R}$ be a GTS and $\mathscr{G}$ a set of graphs which is closed under rule application of $\mathscr{R}$. The *graph transition system* w.r.t. $\mathscr{R}$ and $\mathscr{G}$ is the transition system $\langle \mathscr{G}, \Rightarrow_{\mathscr{R}} \rangle$. A graph transition system $\langle \mathscr{G}, \Rightarrow_{\mathscr{R}} \rangle$ *is of bounded path length* if $\sup_{G \in \mathscr{G}} \ell(G) < \infty$.

Example 1 (GTS of bounded path length)**.** The rules $\langle \varnothing \rightharpoonup (A) \rangle$ and $\langle 1(A) \rightharpoonup 1(A) \to \bullet \rangle$ together with the set of disjoint unions of unboundedly many (possibly non-isomorphic) star-shaped graphs forms a graph transition system of bounded path length.

Remark. Note that we only demand bounded path length. If we additionally demand a bound on the node degree, the number of nodes/edges in each connected component of any graph in the graph class is bounded. This can be shown by an induction over the bound on the path length.

Often we are interested in the predecessors or successors of a given set of states in a transition system.

Definition 7 (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 $\mathrm{pre}^*(S') = \bigcup_{k \geq 0} \mathrm{pre}^k(S')$ 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 $\mathrm{pre}(S)$.

GTSs, when interpreted as TSs, in general have an infinite state space.

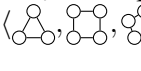
2.3 Well-structuredness

While several problems are undecidable for transition systems in general due to their infinite state space, many interesting decidability results can be achieved if the system is *well-structured* [8, 1, 10].

Definition 8 (well-quasi-order)**.** A *well-quasi-order (wqo)* over a set X is a quasi-order (a reflexive, transitive relation) $\leq \subseteq X \times X$ s.t. every infinite sequence $\langle x_0, x_1, \ldots \rangle$ in X contains an increasing pair $x_i \leq x_j$ with $i < j$.

We give two examples for wqos on graphs. In our setting, the subgraph order is of crucial importance.

Example 2 (subgraph & minor order)**.**

(i) The subgraph order $\leq$ is given by $G \leq H$ iff there is a total injective morphism $G \hookrightarrow H$. Let $\mathscr{G}_\ell$ be a graph class of bounded path length (with bound ℓ). The restriction of $\leq$ to $\mathscr{G}_\ell$ is a wqo [10, 4]. However, it is not a wqo on all graphs: consider, e.g., the infinite sequence $\langle$  $\rangle$ of cyclic graphs of increasing length, which contains no increasing pair.

(ii) The minor order $\preccurlyeq$ is given by $G \preccurlyeq H$ iff G can be obtained from H by a sequence of edge contractions, node and edge deletions. The minor order is a wqo on all graphs [10, 17].

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, depending on the context.

Definition 9 (closure & basis)**.** Let X be a set and $\leq$ a wqo 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. If $A = \uparrow A$, then a *basis* of A is a subset $B \subseteq A$ s.t. (i) B *generates* A, i.e., $\uparrow B = A$, and (ii) any two distinct elements in B are *incomparable*, i.e., $\forall b_1, b_2 \in B : b_1 \neq b_2 \Rightarrow b_1 \not\leq b_2$.

Sets A satisfying $A = \uparrow A$ are later called ideals. For well-structuredness, we demand that the wqo yields a simulation of smaller states by larger states. This condition is called *compatibility*.

Definition 10 (well-structured transition systems)**.** Let $\langle S, \to \rangle$ be transition system and $\leq$ a decidable wqo on S, i.e., for each two given states $s, s' \in S$, it is decidable whether $s \leq s'$. The tuple $\langle S, \leq, \to \rangle$ is a *(strongly) well-structured transition system*, if

(i) The wqo is *(strongly) 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 $s_1' \to^* s_2'$ (strongly: $s_1' \to^1 s_2'$).

(ii) For every $s \in S$, a basis of $\uparrow \mathrm{pre}(\uparrow \{s\})$ is computable.

In Fig. 2, both versions of compatibility are visualized. The term *(strongly) well-structured transition system* is often abbreviated by (S)WSTS. In Sec. 4, we prove the decidability of resilience for SWSTSs. We include the definition of general WSTSs for clarity and to point out the differences. Note that for

(a) Upwards compatibility (b) Strong upwards compatibility

Figure 2: Visualization of the (strong) upwards compatibility property for transition systems.

GTSs, strong compatbility is achieved by applying the same (SPO) rule to bigger graphs. However, in DPO, the bigger graph may not fullfill the dangling condition. Consider, e.g., the rule which deletes a node. This rule can be applied to the graph consisting of a single node but not to the graph ○—○ in DPO.

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

Lemma 1 ([10])**.** *Every graph transition system of bounded path length is strongly well-structured (equipped with the subgraph order).*

Note that in [10], König & Stückrath consider labeled hypergraphs. However, the proof in this case is the same. The premise of bounded path length seems very restrictive, but we can still capture infinitely many graphs. A usual example are graphs where the "topology" remains unchanged. It is also shown in [10] that every *lossy GTS* is well-structured w.r.t. the minor order and without restriction of the graph class. "Lossy" means that every edge contraction rule is contained in the GTS. However, in this case, we do not obtain strong compatibility.

Assumption. In the following, let $\langle S, \leq, \to \rangle$ be a strongly well-structured transition system.

Upward- and downward-closed sets w.r.t. a given wqo are of special interest. Such sets are called ideals and used in Sec. 3 to define resilience problems for WSTSs.

Definition 11 (ideal)**.** An *ideal* $I \subseteq S$ is an upward-closed set, i.e., $\uparrow I = I$. A *bi-ideal* $J \subseteq S$ is an ideal which is also downward-closed, i.e., $\uparrow J = J = \downarrow J$. An *anti-ideal* $J \subseteq S$ is a downward-closed set, i.e., $\downarrow J = J$. The anti-ideal J is decidable if, given $s \in S$, it is decidable whether $s \in J$.

Example 3 (ideal)**.** Let $\mathscr{G}_\ell$ be a graph class of bounded path length. For every positive constraint c, $I_c = \{G \in \mathscr{G}_\ell \mid G \models c\}$ is an ideal.

Bi-ideals often represent "control states" as in [1]. The notion of anti-ideal is the pendent to ideal. Since a downward-closed set does not have an "upward-basis" in general, we will demand that membership is decidable.

Example 4 (anti-ideal)**.** Let $\mathscr{G}_\ell$ be a graph class of bounded path length. For every negative constraint c, $J_c = \{G \in \mathscr{G} \mid G \models c\}$ is a decidable anti-ideal.

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

Fact 2 (stability of ideals)**.** *Let $I, J \subseteq S$ be ideals. Then the sets* $\mathrm{pre}(I)$*, $I \cup J$, and $I \cap J$ are ideals.*

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

Lemma 2 ([1])**.** *For every infinite ascending sequence $\langle I_0 \subseteq I_1 \subseteq \ldots \rangle$ of ideals, there exists a $k \geq 0$ s.t. $I_k = I_{k+1}$. This directly implies $\exists k_0 \geq 0\, \forall k \geq k_0 : I_k = I_{k_0}$.*

Since ideals are in general infinite, we often want a finite representation. Similar to algebraic structures, ideals are represented by a finite basis (a minimal generating set). Indeed, every ideal has a basis and every basis is finite. We consider bases for complexity reasons. In theory, finite generating sets are sufficient to carry out our approach.

Fact 3 ([1])**.** *(i) For every ideal $I \subseteq S$, there exists a finite basis B of I. (ii) Given a finite set $A \subseteq S$ with $I = \uparrow A$, we can compute a finite basis B of I.*

2.4 Ideal Reachability

In [1], Abdulla et al. exploit Lemma 2 to show the decidability of *ideal reachability* (also called *coverability*) for strongly well-structured transition systems. The corresponding algorithm forms the basis of our results. We present its basic idea. For any ideal I, another ideal I^* is constructed, s.t. $\exists s' \in I : s \to^* s'$ iff $s \in I^*$. This is clearly the case for $I^* = \mathrm{pre}^*(I) = \bigcup_{j \geq 0} \mathrm{pre}^j(I)$. The idea is to iteratively construct the sequence of the ideals $I^k = \bigcup_{0 \leq j \leq k} \mathrm{pre}^j(I)$ until it becomes stable.

Definition 12 (index)**.** For an ideal $I \subseteq S$ and $k \geq 0$, let $I^k = \bigcup_{0 \leq j \leq k} \mathrm{pre}^j(I) \subseteq I^{k+1}$. The *index* $k(I)$ is the smallest k_0 s.t. $I^k = I^{k_0}$ for all $k \geq k_0$.

Lemma 2 ensures that $k(I)$ always exists. However, we have to show that $I^k = I^{k+1}$ implies $k(I) \leq k$ to obtain a stop condition. This follows by the observation that $I^{k+1} = I \cup \mathrm{pre}(I^k)$.

Fact 4 (stop condition)**.** *Let $I \subseteq S$ be an ideal and $k \geq 0$ s.t. $I^k = I^{k+1}$, then $I^\ell = I^k$ for all $\ell \geq k$, i.e., $k(I) \leq k$. This also implies that* $\mathrm{pre}^*(I) = I^k$.

Since ideals are infinite, we cannot carry this construction out directly, but we use a basis for representing an ideal. If we can show the computability of a basis in every iteration step, we obtain an algorithm which can decide whether we can reach an ideal I from a given state s.

Lemma 3 ([1])**.** *Given a basis of an ideal $I \subseteq S$, and a state s of a strongly well-structured transition system, we can decide whether we can reach I from s.*

Proof. We have to show that we can compute a basis of I^{k+1} if we are given a basis of I^k. Then the decidability of the stop condition follows directly. Let B be a basis of I^k. We have

$$I^{k+1} = I \cup \mathrm{pre}(I^k) = I \cup \bigcup_{s' \in B} \mathrm{pre}(\uparrow\{s'\}).$$

Since $\mathrm{pre}(\uparrow\{s'\})$ is computable for any $s' \in S$ by definition, we obtain a finite generating set of I^{k+1}. By Fact 3, we can compute a basis of I^{k+1}. □

3 Adverse Conditions and Resilience Problems

We put adverse conditions and resilience into context by using joint graph transformation systems [12]. Abstracting from the setting of GTSs, we identify resilience problems for TSs.

3.1 Joint Graph Transformation Systems

We recapitulate the modeling of adverse conditions by joint graph transformation systems, introduced in [12]. We define joint graph transformation systems, which involve a system and an environment, as well as an automaton modeling the interaction between them. Both, system and environment, are GTSs.

Assumption. In the following, let Λ be a fixed label alphabet, and $\mathscr{S}$ and $\mathscr{E}$ be GTSs over Λ, called *system* and *environment*, respectively. W.l.o.g., we assume that $\mathscr{S}$ and $\mathscr{E}$ are disjoint. (If $\mathscr{S}$ and $\mathscr{E}$ share a common rule r, we assign r different names in $\mathscr{S}$ and $\mathscr{E}$.)

We specify the class of automata which are used to regulate the interaction between system and environment. These control automata are similar to ω-automata, see, e.g., [19].

Definition 13 (control automaton)**.** A *control automaton* of $\langle \mathscr{S}, \mathscr{E} \rangle$ is a tuple $A = \langle Q, q_0, \delta, \mathrm{sel} \rangle$ consisting of a finite set Q disjoint from Λ, called the *state set*, an *initial state* $q_0 \in Q$, a *transition relation* $\delta \subseteq Q \times Q$, and a function $\mathrm{sel} : \delta \to \mathfrak{P}(\mathscr{S} \cup \mathscr{E})$ (into the power set of $\mathscr{S} \cup \mathscr{E}$), called the *selection function*.

A joint graph transformation system is obtained by *synchronizing* the system, repectively, the environment, with the control automaton, and then joining both sets of enriched rules.

Definition 14 (joint graph transformation system)**.** Let $A = \langle Q, q_0, \delta, \mathrm{sel} \rangle$ be a control automaton of $\langle \mathscr{S}, \mathscr{E} \rangle$. The *joint graph transformation system* of $\mathscr{S}$ and $\mathscr{E}$ w.r.t. A is the graph transformation system $\mathscr{S}_A \cup \mathscr{E}_A$ where for a rule set $\mathscr{R} \in \{\mathscr{S}, \mathscr{E}\}$, the *enriched rule set* $\mathscr{R}_A$ is given by

$$\mathscr{R}_A = \{ \langle L, q \rangle \rightharpoonup \langle R, q' \rangle \mid \langle q, q' \rangle \in \delta \text{ and } \langle L \rightharpoonup R \rangle \in \mathscr{R} \cap \mathrm{sel}\langle q, q' \rangle \},$$

and for a graph G and a state q, the tuple $\langle G, q \rangle$ denotes the disjoint union of G and a node labeled with q. In the partial morphism $\langle L, q \rangle \rightharpoonup \langle R, q' \rangle$, the node labeled with q is mapped to the node labeled with q'.

We refine our notion of joint graph transformation systems, namely to *annotated joint graph transformation systems*, which also carry the information whether the last applied rule was a system or environment rule. This is realized by a node labeled with "**s**" or "**e**".

Notation. For a joint graph transformation system $\mathscr{S}_A \cup \mathscr{E}_A$, the symbol $\mathrm{m}(\mathscr{S}) = \mathbf{s}$ or $\mathrm{m}(\mathscr{E}) = \mathbf{e}$, is the *marker* of $\mathscr{S}$ or $\mathscr{E}$, respectively. For a rule $r \in \mathscr{R}$ and $\mathscr{R} \in \{\mathscr{S}, \mathscr{E}\}$, let $\mathrm{m}(r) = \mathrm{m}(\mathscr{R})$ be the marker of r. The set of all markers $M = \{\top, \mathbf{s}, \mathbf{e}\}$ includes also the symbol $\top$, usually indicating a start graph.

For the explicit construction, we can use *premarkers* to reduce the number of rules. For a more extensive account on this technical detail, consult [12].

Definition 15 (annotated joint graph transformation system)**.** Let $\mathscr{S}_A \cup \mathscr{E}_A$ be a joint graph transformation systems w.r.t. a control automaton A of $\langle \mathscr{S}, \mathscr{E} \rangle$. The *annotated joint graph transformation system* of $\mathscr{S}$ and $\mathscr{E}$ w.r.t. A is $\mathscr{S}'_A \cup \mathscr{E}'_A$, where for a rule set $\mathscr{R} \in \{\mathscr{S}, \mathscr{E}\}$, the *marked rule set* $\mathscr{R}'_A$ is defined as

$$\mathscr{R}'_A = \{ \langle L, q, m \rangle \rightharpoonup \langle R, q', m' \rangle \mid \langle L, q \rangle \rightharpoonup \langle R, q' \rangle \in \mathscr{R}_A, m \in M, m' = \mathrm{m}(\mathscr{R}) \},$$

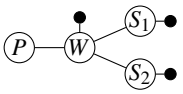
where $\langle G, q, m \rangle$ in turn denotes the disjoint union of a graph G, a node labeled with a state q, and a node labeled with a marker m. In the partial morphism $\langle L, q, m \rangle \rightharpoonup \langle R, q', m' \rangle$, the node lableled with m is mapped to the node labeled with m'.

We explicate the state set of annotated joint GTSs. These graphs are of the form $\langle G, q, m \rangle$ for a state q of the control automaton and a marker m. We denote a class of all such graphs by $\mathscr{G} \oplus Q \oplus M$. Using such graphs instead of the product of graphs we can directly apply the result of [10] for GTSs (Lemma 1).

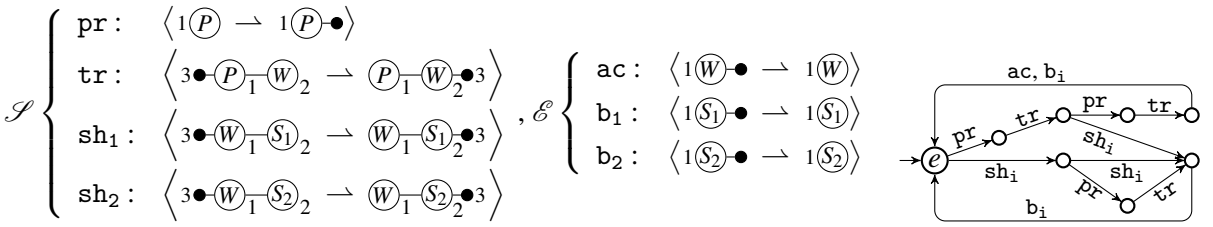
Definition 16 (joint graph transition system)**.** Let $(\mathscr{S}_A \cup \mathscr{E}_A)'$ be an annotated joint GTS and $\mathscr{G}'$ be a class of graphs which is of the form $\mathscr{G} \oplus Q \oplus M$ and closed under rule application of $(\mathscr{S}_A \cup \mathscr{E}_A)'$. The graph transition system $\langle \mathscr{G}', \Rightarrow_{(\mathscr{S}_A \cup \mathscr{E}_A)'} \rangle$ is called *annotated joint graph transition system*.

Note that we usually begin our analysis at a start graph of the form $\langle G, q_0, \top \rangle$.

Example 5 (supply chain)**.** We model a simple supply chain with graph transformation rules. The infrastructure (topology) is given in the following start graph:



A production site (P) is connected to a warehouse (W) which again is connected to two stores S_1 and S_2. Each black node indicates one product at the corresponding (connected) location. The behavior in this production chain is modeled by the graph transformation rules in Fig. 3a. The system rules consists of `pr` (the completion of a product at the production site P), `tr` (transporting a product from P to the warehouse W), and $\mathtt{sh}_1$ and $\mathtt{sh}_2$ (shipping a product from W to one of the two stores S_1, S_2). The environment rules describe external impacts. Namely, `ac` describes an accident in the warehouse which leads to the loss of one product, and $\mathtt{b}_1$ and $\mathtt{b}_2$ describe that a product is bought from S_1 or S_2, respectively. The



(a) The two GTSs $\mathscr{S}$ (system) and $\mathscr{E}$ (environment) (b) Control automaton

Figure 3: A joint GTS consisting of a GTS for system and environment, each, and a control automaton.

control automaton in Fig. 3b describes the possible order of rule applications. We are interested in the question when the product is again *in stock* (at least 1 product in the warehouse and in each of both stores) whenever a customer buys a product or when an accident in the warehouse happens. After each such transition, the automaton is in the state e. Regardless of the current situation, in 17 steps we can accomplish that the product is in stock by first producing and transporting 6 products with a following accident (3 products will get lost) and shipping them to the stores afterwards. However, what is the minimal number of steps in which we can reach a situation where the product is in stock whenever someone bought a product or a product got lost in an accident?

We come back to that question in Ex. 6 in Sec. 5.3. We describe the setting for joint GTSs which we investigate: Consider a safety condition c, given as positive constraint, and the set of graphs $I_c = \{G' \in \mathscr{G} \oplus Q \oplus M \,|\, G' \models c\}$ which satisfy c. Similarly, let $J_{\mathbf{e}} = \{G' \in \mathscr{G} \oplus Q \oplus M \,|\, G' \models \exists \mathbf{e}\}$ (all graphs obtained by an environment interference; $\exists \mathbf{e}$ means that there exists a node labeled with $\mathbf{e}$). The environment is usually modeled in a such way that it has an adverse effect on the satisfaction of c. *Resilience* in this context means that the system can withstand such an adverse condition. We ask whether we can reach a graph in I_c in a reasonable amount of time whenever we reach a graph in $J_{\mathbf{e}}$. By a "reasonable amount of time", we mean either that a number k of steps is given in which I_c should be reached (*explicit resilience*), or that I_c should be reached in a bounded number of steps (*bounded resilience*).

Another approach is to consider the set $J_{\neg c} = \{G' \in \mathscr{G} \oplus Q \oplus M | G' \not\models c\}$ instead of $J_{\mathbf{e}}$. So, we ask whether we can reach a graph which satisfies c in a bounded amount of time/in at most k steps whenever we reach a graph which does not satisfy c, i.e., an error state. Both instances of the problem are reasonable, and if we can give a positive answer for the latter one, we can also give a positive answer for the first one. We focus on the first problem (adverse conditions), but the results we obtain in Sec. 4 abstract from a specific J and therefore also apply to the latter one (error states).

3.2 Abstract Resilience Problems

The previous motivation gives rise to a more abstract definition of resilience problems, namely in the framework of TSs. Recall that, when we explicate a state set, every GTS can be interpreted as a TS.

We assume that a TS $\langle S, \to \rangle$ comes along with a set of propositions each of which is either satisfied or not satisfied by each state of the TS. Let SAFE (*safety* condition) and BAD (*bad* condition) be propositions. Note that BAD is not necessarily equivalent to ¬SAFE. We ask whether we can reach a state which satisfies SAFE in a reasonable amount of time whenever we reach a state which satisfies BAD. From this we formulate two resilience problems. First consider the case where the recovery time is bound by a natural number $k \geq 0$, i.e., the *(abstract) explicit resilience problem*.

EXPLICIT RESILIENCE PROBLEM

Given: A state s of a TS $\langle S, \to \rangle$, propositions SAFE and BAD, a natural number $k \geq 0$.

Question: $\forall s' \in S : (s' \models \text{BAD} \wedge s \to^* s') \Rightarrow \exists s'' \in S : s' \to^{\leq k} s'' \wedge s'' \models \text{SAFE}$?

If we assume that the transition system yields infinite sequences of transitions, we can express the property to be evaluated in CTL by $s \models \mathbf{AG}(\text{BAD} \to \bigvee_{0 \leq j \leq k} \mathbf{EX}^j \text{SAFE})$. We can also ask whether there exists such a bound k. We call this problem the *(abstract) bounded resilience problem*.

BOUNDED RESILIENCE PROBLEM

Given: A state s of a TS $\langle S, \to \rangle$, propositions SAFE and BAD.

Question: $\exists k \geq 0 \; \forall s' \in S : (s' \models \text{BAD} \wedge s \to^* s') \Rightarrow \exists s'' \in S : s' \to^{\leq k} s'' \wedge s'' \models \text{SAFE}$?

Both problems are undecidable: For SAFE = `false`, resilience is equivalent to reachability of BAD.

4 Decidability Results

Many interesting decidability results can be obtained if we assume that a transition system is well-structured [1, 8, 10]. We formulate the resilience problems from the previous section for WSTSs and show decidability of both, the explicit and the bounded resilience problem, in the setting of SWSTSs.

4.1 Resilience Problems in a Well-structured Framework

Properties in well-structured transition systems are often given as upward- or downward closed sets [1, 8]. Ideals enjoy suitable features for verification such as finite representation and stability, and anti-ideals are their complements (cp. Sec. 2.3). Transfering the abstract resilience problems into this framework, it is therefore reasonable to demand that both propositions, SAFE and BAD, are given by ideals or anti-ideals. For our purpose, the following setting suits very well: we assume that the safety property is given by an ideal and the bad condition by a decidable anti-ideal.

From these considerations, we formulate "instances" of the abstract resilience problems for well-structured transition systems. Again, we first consider the case where the recovery time is bounded by a $k \in \mathbb{N}$, the *explicit resilience problem for WSTSs*.

EXPLICIT RESILIENCE PROBLEM FOR WSTSS

Given: A state s of a WSTS $\langle S, \leq, \to \rangle$, a basis of $\uparrow \text{post}^*(s)$, an ideal I with a given basis, a decidable anti-ideal J, a natural number $k \geq 0$.

Question: $\forall s' \in J : (s \to^* s') \Rightarrow \exists s'' \in I : s' \to^{\leq k} s''$?

Analogously, we formulate the *bounded resilience problem for WSTSs*.

BOUNDED RESILIENCE PROBLEM FOR WSTSS

Given: A state s of a WSTS $\langle S, \leq, \rightarrow \rangle$, a basis of $\uparrow \text{post}^*(s)$, an ideal I with a given basis, a decidable anti-ideal J.

Question: $\exists k \geq 0 \; \forall s' \in J : (s \rightarrow^* s') \Rightarrow \exists s'' \in I : s' \rightarrow^{\leq k} s''$?

From now on, we mean one of the previously defined resilience problems for WSTSs if we speak of a resilience problem. If the answer of the bounded (explicit) resilience problem is positive, we say that $\langle S, \leq, \rightarrow \rangle$ is *resilient (k-step resilient)* w.r.t. I and J starting from s. In this context, s is a *start state*.

Remark. The premise that a basis of $\uparrow \text{post}^*(s)$ is given is a strong but reasonable assumption. In general, we cannot simply compute the sequence of ideals $P_k = \bigcup_{0 \leq j \leq k} \uparrow \text{post}^j(s)$ until it becomes stationary. This sequence does become stationary by Lemma 2. However, in contrast to the case in Lemma 3, $P_{k+1} = P_k$ is not a sufficient stop condition. So, this way it is not algorithmically checkable when we have reached k_0 s.t. $P_\ell = P_{k_0}$ for every $\ell \geq k_0$. However, we investigate resilience of GTSs each of which constitutes a SWSTS. A sufficient condition for strong well-structuredness is boundedness of the path length (cp. Lemma 1). This holds, e.g., for graph classes where the "topology" is static. For these graph classes, a basis of all successors is often easier to determine than in general. A typical example for such GTSs are Petri nets, where such a basis is computable (Sec. 5.3). In Sec. 5.2, we drop the assumption, and show that we can still approximate a basis of $\uparrow \text{post}^*(s)$ to achieve approximation results for resilience.

4.2 Decidability

Abdulla et al. show in [1] that ideal reachability is decidable for SWSTSs (cp. Lemma 3). In [8], Finkel & Schnoebelen show that ideal reachability (or *coverability*) is also decidable for WSTSs. Both algorithms coincide in the case of strong well-structuredness. König & Stückrath [10] use the algorithm of [8] for the *backwards analysis* for (generalized) well-structured GTSs.

The main difference between the algorithms in [1] and [8] is that for (not necessarily strongly) WSTSs, $\text{pre}(I')$ in general, for any ideal I', is not an ideal. Thus, Finkel & Schnoebelen consider in every iteration step the ideal $\uparrow \text{pre}(I')$ instead of $\text{pre}(I')$. Now the same arguments like before hold (cp. Sec. 2.4) and a basis of $\text{pre}^*(I) = \uparrow \text{pre}^*(I)$ for a given ideal I can be computed.

We are interested in the exact number of steps which we need to reach an ideal. Thus, $\text{pre}(I')$ should be an ideal and we cannot use the technique from [8] for WSTSs. We need to restrict our setting to *strongly* WSTSs like in [1]. First, we state our main result for SWSTSs, the decidability of resilience.

Theorem 1 (decidability of resilience)**.** *The explicit and the bounded resilience problem both are decidable for strongly well-structured transition systems.*

We prove this theorem by giving a respective algorithm. It exploits a modified version of the ideal reachability algorithm in [1] (cp. Lemma 3). We check in every iteration step inclusion in $I^k = \bigcup_{0 \leq j \leq k} \text{pre}^j(I)$. Before doing so, we need a finite representation of $\text{post}^*(s) \cap J$ to check the inclusion in an ideal I'. The next lemma uses that J and I' are downward- and upward-closed, respectively.

Lemma 4 (intersection with anti-ideal)**.** *Let $A \subseteq S$ be a set, $J \subseteq S$ an anti-ideal and $I' \subseteq S$ an ideal. Then $A \cap J \subseteq I' \Leftrightarrow (\uparrow A) \cap J \subseteq I'$.*

This lemma enables us to prove Thm. 1 given above. We iteratively determine the minimal k satisfying $\text{post}^*(s) \cap J \subseteq I^k$ (or stop, if there does not exist such k).

Proof of Theorem 1. Let B_{post} be a basis of $\uparrow \text{post}^*(s)$, B_0 a basis of I, and J a decidable anti-ideal. For every $k \geq 0$, I^k is an ideal due to strong compatibility. By applying Lemma 4 twice, we obtain

$$\text{post}^*(s) \cap J \subseteq I^k \;\; \Leftrightarrow \;\; B_{\text{post}} \cap J \subseteq I^k$$

for any $k \geq 0$. Since B_{post} is finite and J is a decidable anti-ideal, we can directly compute $B_{\text{post}} \cap J$. We perform a modification of the ideal reachability algorithm: Iteratively check whether $B_{\text{post}} \cap J \subseteq I^k$. If this is the case, return $k_{\min} = k$. Otherwise check whether $I^{k+1} = I^k$. If so, return -1 (`false`), otherwise continue. We have to make sure that every iteration step is decidable. In fact, we can compute a basis of I^{k+1} if we have a basis of I^k. This follows by the proof of Lemma 3. The stop condition is decidable and by Fact 4 also sufficient. Soundness and completeness follow by the previous considerations and the fact that

$$\text{post}^*(s) \cap J \subseteq I^k \quad \Leftrightarrow \quad (\forall s' \in J : (s \to^* s') \Rightarrow \exists s'' \in I : s' \to^{\leq k} s'')$$

for any $k \geq 0$. Termination is guaranteed by Lemma 2.

To sum up, our algorithm decides whether there exists a $k \geq 0$ s.t. $\text{post}^*(s) \cap J \subseteq I^k$, and returns the minimal such k in the positive case. Thus, it decides the bounded resilience problem. Given any k, we can check whether $k_{\min} \leq k$ and therefore decide the explicit resilience problem. □

We denote the above described algorithm deciding resilience by MINIMALSTEP(B_{post}, J, B_0) and the used procedure returning a basis of $\text{pre}(\uparrow B')$ by PREBASIS(B'). It is shown in [10], that such a prebasis is computable for GTSs, and described in detail in [18]. The method MIN(B') minimizes a finite set B' by deleting every element in B' for which there is already a smaller element in B'.

Algorithm 1 Minimal k Algorithm

```
1:  procedure MINIMALSTEP(B_post, J, B_0)        ▷ k_min (minimal upper bound for recovery time)/−1
2:      B ← B_post ∩ J                           ▷ compute B by taking out elements which are not in J
3:      k ← 0                                    ▷ increasing counter
4:      B_1 ← B_0                                ▷ basis of the current I^k; B_0 is a given basis of I
5:      B_2 ← ∅                                  ▷ basis of the current I^{k+1}
6:      while true do
7:          if B ⊆ ↑B_1 then
8:              return k                         ▷ we found k_min
9:          else
10:             B_2 ← B_0 ∪ PREBASIS(B_1)        ▷ PREBASIS(B_1) computes the basis of ↑B_1
11:             B_2 ← MIN(B_2)                   ▷ MIN(B_2) minimizes the set B_2
12:             if B_2 ⊆ ↑B_1 then
13:                 return −1                    ▷ there exists no such k
14:             else
15:                 B_1 ← B_2                    ▷ continue
16:                 k ← k + 1
17:             end if
18:         end if
19:     end while
20: end procedure
```

In the proof of Thm. 1, it was crucial that we have *strong* compatibility. This approach does not work for WSTSs in general. We loose precision when we only demand compatibility. Thus, we conjecture that both resilience problems are undecidable for WSTSs in general, but this question remains still open.

5 Application to Graph Transformation Systems

We apply the abstract results of the previous section to (joint) graph transformation systems and present a framework for verifying resilience of GTSs. We exemplarily show how Petri nets fit in this setting and give also an example beyond Petri nets.

We considered ideals as safety, and decidable anti-ideals as "bad" conditions. In the setting of well-structured GTSs w.r.t. the subgraph order, these can be expressed as positive and negative constraints. Recall that, for a fixed class $\mathscr{G}$ of graphs, $I_c = \{G \in \mathscr{G} \mid G \models c\}$ for a positive constraint c, and $J_{c'} = \{G \in \mathscr{G} \mid G \models c'\}$ for a negative constraint c'.

Fact 5 (ideals of graphs)**.** *Let $\mathscr{G}_\ell$ be a class of graphs of bounded path length. Let $I, J \subseteq \mathscr{G}_\ell$ be sets.*

(i) I is an ideal $\Leftrightarrow I = I_c$ for a positive constraint c.

(ii) J is a decidable anti-ideal $\Leftrightarrow J = J_c$ for a negative constraint c.

Thus, for GTSs, our safety conditions are equivalent to positive constraints and bad conditions are equivalent to negative constraints.

Remark. More general graph constraints, e.g., $\forall(\,\overset{1}{\circ}\,, \exists(\overset{}{\circ}_1))$, do not constitute ideals w.r.t. the subgraph order. The relation $\overset{}{\circ}_1 \leq \overset{}{\circ}_1\ \circ$ shows that upward-closedness is not guarenteed. In special cases, (nested) graph constraints [15, 9] may yield ideals, e.g., the ideal in the later discussed Ex. 7 can be expressed as $\forall(1\textcircled{L}\ \textcircled{L}2, \exists(1\textcircled{L}\to\bullet\to\textcircled{L}2))$. However, we conjecture that a generalization to more arbitrary (nested) graph contraints is not possible.

5.1 Verifying Resilience of Graph Transformation Systems

Using the sufficient conditions for strong well-structuredness of König & Stückrath [10], we obtain the decidability of both resilience problems for a subclass of GTSs. We need to use the subgraph order as wqo. Thus, we have the restriction of bounded4 path length for the considered graph class. Instead of considering GTSs, we consider graph transition systems, i.e., we always explicate the state set. Thm. 1 and the result in [10] (see Lemma 1) imply our main result for GTSs:

Theorem 2 (decidability of resilience for well-structured GTSs)**.** *The explicit and the bounded resilience problem are decidable for graph transition systems which are of bounded path length (and equipped with the subgraph order).*

As joint GTSs are also GTSs, the same sufficient conditions for strong well-structuredness apply.

Fact 6 (strongly well-structured joint GTSs)**.** *Every annotated joint graph transition system which is of bounded path length is strongly well-structured (equipped with the subgraph order).*

An immediate consequence of Thm. 2 and Fact 6 is the following:

Corollary 1 (decidability of resilience for joint GTSs)**.** *The explicit and the bounded resilience problem are decidable for annotated joint graph transition systems which are of bounded path length (and equipped with the subgraph order).*

Thus, we can apply the algorithm MINIMALSTEP described in Sec. 4.2 to verify resilience of annotated joint graph transition systems. We consider an ideal I_c for a positive constraint c with a given basis B_c. The anti-ideal (bi-ideal) is given by $J_{\mathbf{e}} = \{G' \in \mathscr{G} \oplus Q \oplus M \mid G' \models \exists \mathbf{e}\}$. We assume that a start graph $G \in \mathscr{G} \oplus \{q_0\} \oplus \{\top\}$ and a basis B_G of $\uparrow \mathrm{post}^*(G)$ are given. The PREBASIS procedure for the subgraph order needed in the algorithm is given by König & Stückrath in [10] (and more detailed in [18]).

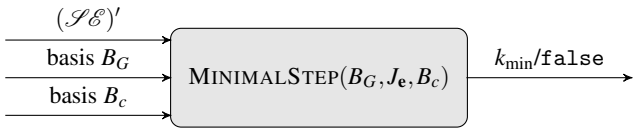


Figure 4: Verifying resilience in the adverse conditions approach.

5.2 Approximations

We now drop an essential assumption for the decidability results in Sec. 4.2 by considering SWSTSs without a given basis of $\uparrow \mathrm{post}^*(s)$. We show that we can still approximate $k_{\min}$ from below (by k^ℓ_{un}, $\ell \in \mathbb{N}$) and above (by k_{ov}) by calculating corresponding approximations of (a basis of) $\uparrow \mathrm{post}^*(s)$. The following function, called μ-function, defines these approximations.

Definition 17 (μ-function, k^ℓ_{un}, k_{ov})**.** Let $\langle S, \leq, \rightarrow \rangle$ be a SWSTS, $J \subseteq S$ an anti-ideal, and $I \subseteq S$ an ideal. We define the function $\mu : \mathfrak{P}(S) \to \mathbb{N} \cup \{\infty\}$ as $\mu(A) = \min(\{k \in \mathbb{N} : A \cap J \subseteq \bigcup_{j \leq k} \mathrm{pre}^j(I)\} \cup \{\infty\})$ where $\mathfrak{P}(S)$ is the power set of S. For $s \in S$ and $\ell \in \mathbb{N}$, let $k^\ell_{\mathrm{un}} := \mu(\bigcup_{j \leq \ell} \mathrm{post}^j(s))$ and $k_{\mathrm{ov}} := \mu(\mathrm{post}^*(\uparrow\{s\}))$.

Note that $k_{\min} = \mu(\mathrm{post}^*(s))$ and that $k_{\min} = \infty$ can be read as "there is no such k". By definition, μ is monotonic, i.e., $A \subseteq B$ implies $\mu(A) \leq \mu(B)$, and by Lemma 4, $\mu(\uparrow A) = \mu(A)$. For the under- and over-approximation, we consider a basis of $\uparrow \bigcup_{j \leq \ell} \mathrm{post}^j(s)$ and a basis of $\uparrow \mathrm{post}^*(\uparrow\{s\})$, respectively. For every GTS of bounded path length, this under-approximation is feasible. We present an idea for performing the over-approximation by means of invertibility.

Fact 7 (weak invertibility)**.** *Let $\langle \mathscr{G}, \Rightarrow_{\mathscr{R}} \rangle$ be a graph transition system of bounded path length and $\mathscr{R}'$ a GTS s.t. $G \Rightarrow^*_{\mathscr{R}} H$ iff $H \Rightarrow^*_{\mathscr{R}'} G$ for all $G, H \in \mathscr{G}$. Then, for every $G \in \mathscr{G}$, $\mathrm{post}^*_{\mathscr{R}}(\uparrow\{G\}) = \mathrm{pre}^*_{\mathscr{R}'}(\uparrow\{G\})$ and a basis of $\mathrm{post}^*_{\mathscr{R}}(\uparrow\{G\})$ is computable.*

In particular, such an $\mathscr{R}'$ exists if $G \Rightarrow_r H$ iff $H \Rightarrow_{r^{-1}} G$ for all $G, H \in \mathscr{G}$, $r \in \mathscr{R}$, where for a rule $r = \langle L \rightharpoonup R \rangle$ which is injective on its domain, $r^{-1} = \langle R \rightharpoonup L \rangle$ is the *inverse rule*. In general, $G \Rightarrow_r H$ only implies that there is a graph $G' \leq G$ s.t. $H \Rightarrow_{r^{-1}} G'$, since an application of r may have deleted dangling edges. However, in some classes of GTSs, e.g., in Petri nets (see Sec. 5.3), there are no dangling edges in both directions, and we can use the inverse rules for the over-approximation.

Fact 8 (approximation)**.** *Let $\langle \mathscr{G}, \Rightarrow_{\mathscr{R}} \rangle$ be a GTS of bounded path, $J \subseteq \mathscr{G}$ an anti-ideal, $I \subseteq \mathscr{G}$ an ideal, and $G \in \mathscr{G}$. (i) For every $\ell \geq 0$, k^ℓ_{un} is computable and $k^\ell_{un} \leq k_{\min}$. The sequence $\langle k^\ell_{un} \rangle_\ell$ converges to $k_{\min}$, eventually stabilizing. (ii) Under the assumptions of Fact 7, k_{ov} is computable and $k_{\mathrm{ov}} \geq k_{\min}$.*

Note that $k^\ell_{\mathrm{un}} = \infty$ implies $k_{\min} = \infty$, and $k_{\mathrm{ov}} < \infty$ implies $k_{\min} < \infty$. Only if $k^\ell_{\mathrm{un}} = 0$ and $k_{\mathrm{ov}} = \infty$, we gain no information about $k_{\min}$. The approximation results described above are visualized in Fig. 5.

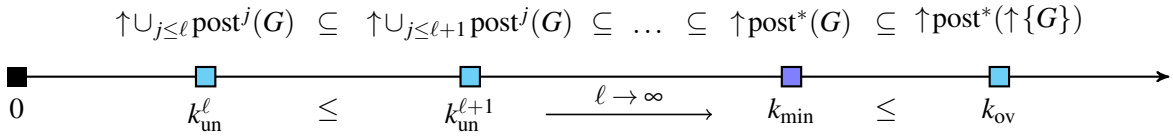


Figure 5: Under- and over-approximation of $k_{\min}$ by corresponding approximation of $\uparrow \mathrm{post}^*(G)$.

5.3 An Example Class: Petri Nets

Petri nets [14] are a common model for discrete distributed systems in computer science, often applied, e.g., in logistics or supply chains [22]. It is a classical example for strongly well-structured (graph) transition systems. We will give a definition of Petri nets and show how our example fits in this setting.

Definition 18 (Petri nets)**.** A *Petri net* is a tuple $N = \langle P, T, F \rangle$ with disjoint finite sets of *places* P and *transitions* T, and a *flow function* $F : (P \times T) \cup (P \times T) \to \mathbb{N}$. A *marking* in N is a multi-set $M : P \to \mathbb{N}$ that indicates the number of tokens on each place. $F(x,y) = n > 0$ means there is an *arc* of *weight* n from node x to y describing the flow of tokens in the net. A transition $t \in T$ is *enabled* in a marking M if $\forall p \in P : F(p,t) \leq M(p)$. If t is enabled, then t can *fire* in M, leading to a new marking M' calculated by $\forall p \in P : M'(p) = M(p) - F(p,t) + F(t,p)$. This is denoted by $M[t\rangle M'$. Usually, a Petri net N is equipped with an *initial marking* M_0. The tuple $\langle N, M_0 \rangle$ is then called a *marked Petri net*.

Any Petri net N can be interpreted as a transition system with the states S given by $\mathscr{M}(N)$, the set of all markings of N, and the transitions $\to$ given by $M \to M' \Leftrightarrow \exists t \in T : M[t\rangle M'$. Together with the wqo $\leq_{\text{PN}}$, given by $\forall M, M' \in \mathscr{M}(N) : M \leq_{\text{PN}} M' :\Leftrightarrow \forall p \in P : M(p) \leq M'(p)$, this constitutes a SWSTS. For Petri nets, reachability and equivalent problems are decidable [14, 7]. From this fact and the results in [21], one can show that for Petri nets a basis of $\uparrow \text{post}^*(M_0)$ is computable: In [21], it is shown that for any ideal I of markings in a Petri net, a basis of I is computable iff for every ω-marking M it is decidable whether $I \cap \downarrow\{M\} = \varnothing$. An ω-marking is a function $M : P \to \mathbb{N} \cup \{\omega\}$, and analogously to before, $\downarrow\{M\} := \{M' \in \mathscr{M}(N) \,|\, \forall p \in P : M'(p) \leq M(p) \vee M(p) = \omega\}$. Since $\uparrow \text{post}^*(M_0)$ is an ideal, we can apply this result and ask whether $\uparrow \text{post}^*(M_0) \cap \downarrow\{M\} = \varnothing$ is decidable. This is obviously equivalent to $\uparrow \text{post}^*(M_0) \cap \downarrow\{M\} \subseteq \varnothing$, allowing us to apply Lemma 4, since $\varnothing$ is an ideal. Thus, we now ask whether

$$\text{post}^*(M_0) \cap \downarrow\{M\} = \varnothing.$$

This problem corresponds to the so-called *submarking reachability problem*, which is decidable (cp., e.g., [7]), since it is recursively equivalent the to reachability problem. Therefore, we get that a basis of $\uparrow \text{post}^*(M_0)$ is computable.

Petri nets can also be seen as an instance of GTSs, as shown in [2]. From that point of view, every transition corresponds to a graph transformation rule. A marking is given by the structure of the Petri net represented as a graph, with the number of tokens on a place represented by extra nodes connected to it, as in Fig. 1c. The wqo $\leq_{\text{PN}}$ then directly corresponds to the subgraph order. Together with the start graph representing the initial marking, interpreting the GTS as a WSTS results in exactly the same SWSTS above. This means we can apply the algorithm deciding resilience in GTS to Petri nets. We demonstrate this by the following example, where we consider a Petri net that, when interpreted as a GTS, is exactly the supply chain modeled in Ex. 5.

Example 6 (supply chain as Petri net)**.** We consider a marked Petri net modeling a simplified scenario of a supply chain, shown in Fig. 6. As usual we depict places as circles, transitions as rectangles, and the flow as weighted directed arcs between them. In the example, all weights are 1 and therefore not indicated. Dots on places indicate the number of tokens on the respective place in the initial marking.

The Petri net corresponds directly to the graph transformation rules in Ex. 5, with the blue transitions simulating $\mathscr{S}$, and the red (checkered) transitions simulating $\mathscr{E}$. The initial marking represents the start graph. Correspondingly, the control automaton has the same structure as in Ex. 5, with transitions replacing rules. Let $I = \{\langle M, q \rangle \,|\, M(\mathit{warehouse}), M(\mathit{store}_1), M(\mathit{store}_2) \geq 1 \wedge q \in Q\}$, i.e., in the warehouse and in both stores products are available for shipping or purchase, respectively. The transitions corresponding to $\mathscr{E}$ reduce the number of tokens in the net. We consider the resilience problem with adverse conditions. By definition of the control automaton, we know that $J_{\mathbf{e}} = \{\langle M, e \rangle | M \text{ is a marking}\}$.

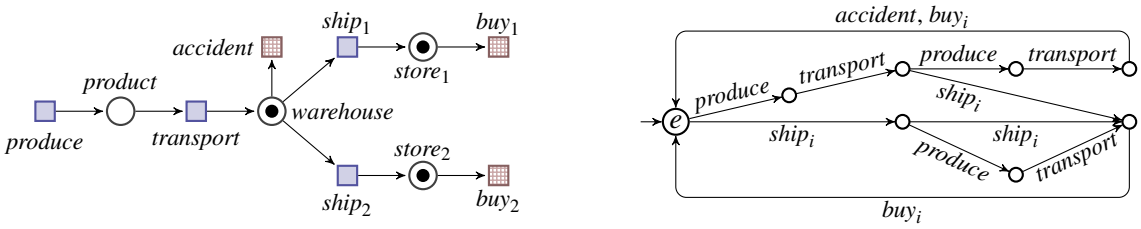


Figure 6: A Petri net modeling a supply chain, and its control automaton.

We interpreted Ex. 5/ Ex. 6 as joint GTS and applied a prototype implementation of the algorithm MINIMALSTEP from Sec. 4.2 to it. We obtained that $k_{\min} = 6$ is the smallest k for which the system is k-step resilient: The following set $B_{\mathbf{e}}^{M_0}$ is the intersection of a basis of $\uparrow \mathrm{post}^*(M_0)$ with $J_{\mathbf{e}}$ where $M_0 = \langle 0,1,1,1,q_0\rangle$. The first coordinate corresponds to (the number of tokens in) *P*/*product*, the second coordinate to *W*/*warehouse*, and the third and fourth coordinate correspond to S_1/$store_1$ and S_2/$store_2$, respectively.

$$B_{\mathbf{e}}^{M_0} = \{\langle 0,1,1,1\rangle, \langle 0,5,0,0\rangle, \langle 0,0,3,0\rangle, \langle 0,0,0,3\rangle, \langle 0,1,2,0\rangle,$$
$$\langle 0,1,0,2\rangle, \langle 0,0,2,1\rangle, \langle 0,0,1,2\rangle, \langle 0,3,1,0\rangle, \langle 0,3,0,1\rangle\} \times \{q_0\}$$

We computed B^k, a basis of I^k, for $1 \le k \le 21$. We only give $B^k \cap J_{\mathbf{e}}$ for $1 \le k \le 6$:

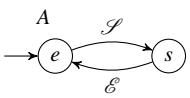
$$B^1 \cap J_{\mathbf{e}} = \{\langle 0,1,1,1\rangle, \langle 0,2,0,1\rangle, \langle 0,2,1,0\rangle\} \times \{q_0\}$$
$$B^2 \cap J_{\mathbf{e}} = \{\langle 0,0,1,1\rangle, \langle 0,2,0,1\rangle, \langle 0,2,1,0\rangle, \langle 0,3,0,0\rangle\} \times \{q_0\}$$
$$B^3 \cap J_{\mathbf{e}} = \{\langle 0,0,1,1\rangle, \langle 0,1,0,1\rangle, \langle 0,1,1,0\rangle, \langle 0,3,0,0\rangle\} \times \{q_0\}$$
$$B^4 \cap J_{\mathbf{e}} = B^5 \cap J_{\mathbf{e}} = B^3 \cap J_{\mathbf{e}}$$
$$B^6 \cap J_{\mathbf{e}} = \{\langle 0,0,1,1\rangle, \langle 0,1,0,1\rangle, \langle 0,1,1,0\rangle, \langle 0,3,0,0\rangle, \langle 0,0,2,0\rangle, \langle 0,0,0,2\rangle\} \times \{q_0\}$$

We obtain $B_{\mathbf{e}}^{M_0} \not\subseteq \uparrow B^k \cap J_{\mathbf{e}}$ for $1 \le k \le 5$, but $B_{\mathbf{e}}^{M_0} \subseteq \uparrow B^6 \cap J_{\mathbf{e}}$. Thus, $k_{\min} = 6$.

5.4 An Example beyond Petri Nets

We give an example for a joint GTS which cannot be modeled by a (finite) Petri net and verify its resilience.

Example 7 (path game)**.** Consider two fixed locations represented by nodes labeled with L. Points between them are represented by black nodes. The system tries to construct two directed paths of length 2 between the locations, one path forth and one back, using the rules $\mathscr{S}$ in Fig. 7. The respective ideal is therefore given by $\exists(\textcircled{L}\!\!\bullet\!\!\textcircled{L}) \vee \exists(\textcircled{L}\!\!\bullet\!\!\bullet\!\!\textcircled{L})$. The environment deletes edges in the graph, corresponding to $\mathscr{E}$ in Fig. 7. The control automaton is alternating:



Thus, one may consider this as a game with alternating turn order. The system can (i) create a new middle point connected to the locations by the rule `New`, (ii) create two parallel edges provided that there

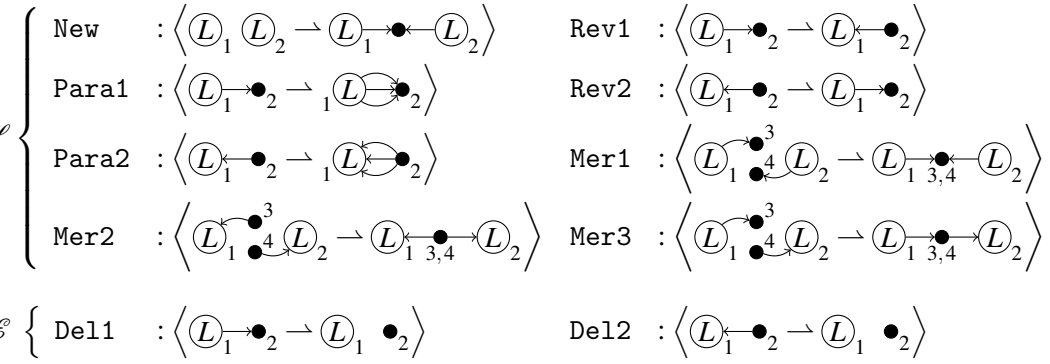
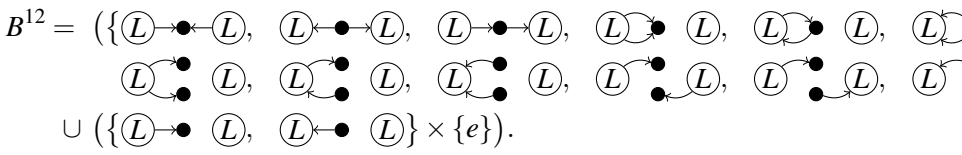


Figure 7: Components of the joint GTS

is one by the `Para`-rules, (iii) reverse the direction of an edge by the `Rev`-rules, and (iv) merge two middle points each of which are connected to a different location by the `Mer`-rules. We ask whether the system can construct the two directed paths of length 2 in a bounded number of rounds (steps) when the environment made its turn, regardless of the current situation. If so, what is the minimal number of steps?

We can reach the graph $G_{LL} := \textcircled{L}\ \textcircled{L}$ (modulo isolated nodes) when the system is only changing the direction of edges. Hence, $\langle G_{LL}, e\rangle \in \uparrow \mathrm{post}^*(G) \cap J_{\mathbf{e}}$ for any start graph G with exactly two locations, arbitrarily many middle points, and arbitrary edges between middle points and locations. Therefore, we only check when $\langle G_{LL}, e\rangle$ occurs the first time in a basis B^k. We applied a prototype implementation of the algorithm in Sec. 4.2 to this example and obtained $B^{13} \cap J_{\mathbf{e}} = \{\langle G_{LL}, e\rangle\} \not\subseteq B^{12}$ by computation of



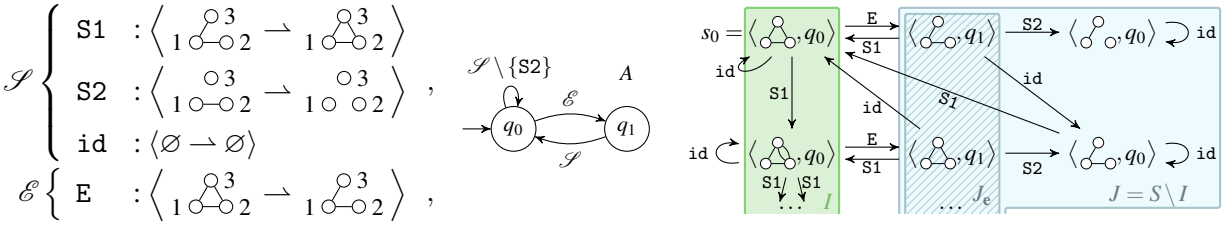
Thus, $k_{\min} = 13$.

Note that we consider equivalence classes of graphs modulo isolated middle points. This has no effect on the well-structuredness of this example. Also note that leaving out the rules for merging has only a slight impact on the bases and no effect on $k_{\min}$.

5.5 Adverse Conditions vs. Error States

We compare the adverse conditions approach with the error state approach. As pointed out, these two views of resilience are not equivalent. While every system that is resilient w.r.t. error states (i.e., $J = S \setminus I$) is also resilient w.r.t. adverse conditions (i.e., $J = J_{\mathbf{e}}$) due to $J_{\mathbf{e}} \setminus I \subseteq S \setminus I$ (meaning that if we can reach I from *every* state, then also from every state in $J_{\mathbf{e}}$), the opposite does not hold in general.

We do not define a restriction on the system/environment to allow more freedom of modeling but our counterexample in Fig. 8 captures the adverse effect of the environment. The joint GTS in Fig. 8a, together with a start graph  , results in the state set in Fig. 8b. A basis of I is given by $\langle$$, q_0\rangle$. From the definition of A, we see that $J_{\mathbf{e}}$ is given by $\{\langle G, q_1\rangle \mid$  $\leq G\}$, indicated by the hatched area. We see that from every reachable state in $J_{\mathbf{e}}$ we can reach I in one step, which means that the system is 1-step resilient w.r.t. adverse conditions. On the other hand, we cannot reach I from the state $\langle$  $, q_0\rangle \in S \setminus I$,



(a) Components of the joint GTS

(b) The state set of the joint GTS

Figure 8: A joint GTS example that is 1-step resilient w.r.t. $J_{\mathbf{e}}$ (adverse conditions), but not resilient w.r.t. $J = S \setminus I$ (error states).

which is reachable from $J_{\mathbf{e}}$ when the "wrong" system rule is applied. This means the system is *not* resilient w.r.t. error states.

If, due to the structure of a joint GTS, we can reach $J_{\mathbf{e}}$ from every reachable error state, as, e.g., in Ex. 5, both approaches coincide. The computed $k_{\min}$'s then only differ by at most the index $k(J_{\mathbf{e}})$.

6 Related Work

We use SPO graph transformation for modeling systems as in **Löwe** [11] (see also **Ehrig et al.** [6]).

Our notion of joint GTSs is a special case of graph-transformational interacting systems. Another approach considering dependencies can be found, e.g., in **Corradini et al.** [3].

The concept of resilience is broadly used in different areas, e.g., in industrial control systems [20, 16], with varying definitions. Following these ideas, we formulated resilience in the abstract settings of TSs and GTSs. Our interpretation of resilience captures recovery in bounded time.

Abdulla et al. [1] show the decidability of ideal reachability (coverability), eventuality properties and simulation in (labeled) SWSTSs. We use the presented algorithm to show the decidability of resilience problems in SWSTSs.

Finkel & Schnoebelen [8] show that the concept of well-structuredness is ubiquitous in computer science by providing a large class of example models (e.g., Petri nets and their extensions, communicating finite state machines, lossy systems, basic process algebras). Moreover, they give several decidability results for systems with different degrees of well-structuredness. They also generalize the algorithm of [1] to (not necessarily strongly) WSTSs to show decidability of coverability.

In [10], **König & Stückrath** extensively study the well-structuredness of GTSs. More detailed considerations can be found in [18]. They identify three types of wqos (minor, subgraph, induced subgraph) on graphs based on results of **Ding** [4] and **Robertson & Seymour** [17]. The fact that the subgraph order is a wqo on graphs of bounded path length while the minor order allows all graphs comes with a trade-off: For obtaining well-structuredness w.r.t. the minor order, the GTS must contain all edge contraction rules, i.e., it must be a "lossy" GTS. On the other hand, all GTSs (without application conditions) 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 (in particular, we use the pred-basis procedure in the case of the subgraph order for our algorithm). In our setting, the regarded wqo is the subgraph order since it yields strong compatibility. They also generalize the notion of well-structured transition systems by regarding Q-restricted WSTSs whose state sets needs not to be a wqo but rather a subset Q of the states is a wqo. König & Stückrath develope a backwards algorithm based on [8] for Q-restricted WSTSs obtaining decidability of coverability under additional assumptions. For SWSTSs, this approach coincides with the

ideal reachability algorithm [1].

All in all, our result for SWSTSs uses a modification of Abdulla et. al [1], and our application to GTSs additionally uses the predecessor-basis procedure from König & Stückrath [10] in every computation step. It can also be seen as a modification of the backwards analysis of König & Stückrath [10] in the case of the subgraph order. We summarize the relations of our results and the used concepts in Fig. 9.

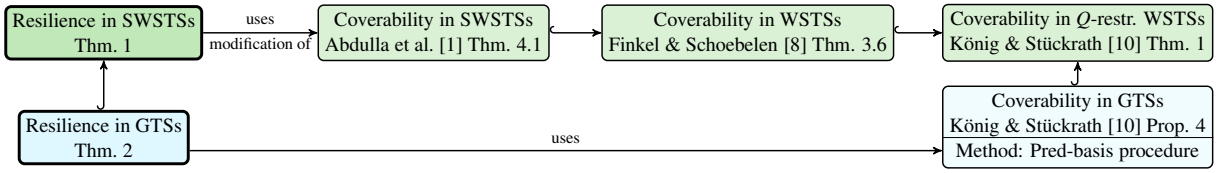


Figure 9: Relation of the decidability results for resilience (bold) and the results in related work. The bottom (blue) and the top (green) layers contain decidability results for GTSs and WSTSs, respectively. The hooked arrows (↪) mean "generalized to" or "instance of".

7 Conclusion

We provided a definition of resilience in an abstract framework, namely the explicit and the bounded resilience problem, and proved decidability of both problems for strongly well-structured transition systems. By application of this theory, we obtained decidability results for GTSs of bounded path length, and in particular a verification framework for GTSs which incorporates adverse conditions.

Our results require that a basis of the upward-closure of all successors is given. Although determining this basis for GTSs is a difficult task, it is computable for Petri nets and can be computed for other GTSs in special cases. We showed how to approximate such a basis when the assumption is dropped, thereby approximating the answer to the resilience problems. In this paper, the used well-quasi-order on graphs is the subgraph order. For the proof, the requirement of *strong* compatibility is crucial. Our approach does not work for lossy GTSs which are well-structured w.r.t. the minor order. We conjecture that both resilience problems are undecidable for lossy GTSs. Ideals w.r.t. the subgraph order can be represented by positive basic graph constraints. In general, nested graph constraints do not constitute ideals.

Future work. We will investigate on (1) the (un)decidability of resilience for WSTSs/lossy GTSs, (2) synthesis of resilient GTSs, i.e., using the presented approach to construct provably resilient GTSs, and (3) the computability of a basis of the upward-closure of all successors for (a subclass of) strongly well-structured GTSs. Regarding (2), we will investigate on the construction of strongly well-structured GTSs. Regarding (3), we will consider further methods for achieving approximation results for resilience.

Acknowledgment. We are grateful to Annegret Habel, Christian Sandmann, and the anonymous reviewers for their helpful comments to this paper. We thank Barbara König for the discussion about approximation and computation of the upward-closure of all successors, and Detlef Plump for the note on graph classes of bounded path length and bounded node degree.

References

[1] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson & Yih-Kuen Tsay (1996): *General Decidability Theorems for Infinite-State Systems*. In: *Proc. LICS 1996*, IEEE Computer Society Press, pp. 313–321,

doi:10.1109/LICS.1996.561359.

[2] Paolo Baldan, Andrea Corradini, Fabio Gadducci & Ugo Montanari (2010): *From Petri Nets to Graph Transformation Systems*. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* 26, doi:10.14279/tuj.eceasst.26.368.

[3] Andrea Corradini, Luciana Foss & Leila Ribeiro (2008): *Graph Transformation with Dependencies for the Specification of Interactive Systems*. In: *Proc. WADT 2008*, *LNCS* 5486, Springer, pp. 102–118, doi:10.1007/978-3-642-03429-9_8.

[4] Guoli Ding (1992): *Subgraphs and well-quasi-ordering*. *J. Graph Theory* 16(5), pp. 489–502, doi:10.1002/jgt.3190160509.

[5] Hartmut Ehrig, Karsten Ehrig, Ulrike Prange & Gabriele Taentzer (2006): *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series, Springer, doi:10.1007/3-540-31188-2.

[6] Hartmut Ehrig, Reiko Heckel, Martin Korff, Michael Löwe, Leila Ribeiro, Annika Wagner & Andrea Corradini (1997): *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*, World Scientific, pp. 247–312, doi:10.1142/9789812384720_0004.

[7] Javier Esparza & Mogens Nielsen (1994): *Decidability Issues for Petri Nets*. *BRICS Report Series* 1(8), doi:10.7146/brics.v1i8.21662.

[8] Alain Finkel & Philippe Schnoebelen (2001): *Well-structured transition systems everywhere!* *Theor. Comput. Sci.* 256(1-2), pp. 63–92, doi:10.1016/S0304-3975(00)00102-X.

[9] Annegret Habel & Karl-Heinz Pennemann (2009): *Correctness of high-level transformation systems relative to nested conditions*. *Math. Struct. Comput. Sci.* 19(2), pp. 245–296, doi:10.1017/S0960129508007202.

[10] Barbara König & Jan Stückrath (2017): *Well-structured graph transformation systems*. *Inf. Comput.* 252, pp. 71–94, doi:10.1016/j.ic.2016.03.005.

[11] Michael Löwe (1991): *Extended algebraic graph transformation*. Ph.D. thesis, Technical University of Berlin, Germany. Available at `http://d-nb.info/910935696`.

[12] Okan Özkan (2020): *Modeling Adverse Conditions in the Framework of Graph Transformation Systems*. In: *Proc. GCM@STAF 2020*, *EPTCS* 330, pp. 35–54, doi:10.4204/EPTCS.330.3.

[13] Subhav Pradhan, Abhishek Dubey, Tihamer Levendovszky, Pranav Srinivas Kumar, William A. Emfinger, Daniel Balasubramanian, William Otte & Gabor Karsai (2016): *Achieving resilience in distributed software systems via self-reconfiguration*. *Journal of Systems and Software* 122, pp. 344–363, doi:10.1016/j.jss.2016.05.038.

[14] Wolfgang Reisig (1985): *Petri Nets: An Introduction*. *EATCS Monographs on Theoretical Computer Science* 4, Springer, doi:10.1007/978-3-642-69968-9.

[15] Arend Rensink (2004): *Representing First-Order Logic Using Graphs*. In: *Proc. ICGT 2004*, *LNCS* 3256, Springer, pp. 319–335, doi:10.1007/978-3-540-30203-2_23.

[16] Craig G. Rieger, Kevin L. Moore & Thomas L. Baldwin (2013): *Resilient control systems: A multi-agent dynamic systems perspective*. In: *Proc. EIT 2013*, IEEE, pp. 1–16, doi:10.1109/EIT.2013.6632721.

[17] Neil Robertson & Paul D. Seymour (2004): *Graph Minors. XX. Wagner's conjecture*. *J. Comb. Theory, Ser. B* 92(2), pp. 325–357, doi:10.1016/j.jctb.2004.08.001.

[18] Jan Stückrath (2016): *Verification of Well-Structured Graph Transformation Systems*. Ph.D. thesis, University of Duisburg-Essen. Available at `https://nbn-resolving.org/urn:nbn:de:hbz:464-20160425-093027-1`.

[19] Wolfgang Thomas (1990): *Automata on Infinite Objects*. In: *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, Elsevier and MIT Press, pp. 133–191, doi:10.1016/b978-0-444-88074-1.50009-3.

[20] Kishor S. Trivedi, Dong Seong Kim & Rahul Ghosh (2009): *Resilience in computer systems and networks*. In: *Proc. ICCAD 2009*, ACM, pp. 74–77, doi:10.1145/1687399.1687415.

[21] Rüdiger Valk & Matthias Jantzen (1985): *The Residue of Vector Sets with Applications to Decidability Problems in Petri Nets*. *Act. Inf.* 21, pp. 643–674, doi:10.1007/BF00289715.

[22] Xiaoling Zhang, Qiang Lu & Teresa Wu (2009): *Petri-net based application for supply chain management: An overview*. In: *Proc. IEEM 2009*, pp. 1406–1410, doi:10.1109/IEEM.2009.5373050.