Probabilistic Graph Transformation Systems

Christian Krause* and Holger Giese

Hasso Plattner Institute at the University of Potsdam
Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany
{christian.krause,holger.giese}@hpi.uni-potsdam.de

Abstract. In the recent years, extensions of graph transformation systems with quantitative properties, such as real-time and stochastic behavior received considerable attention. In this paper, we describe the new quantitative modeling approach of *probabilistic graph transformation systems* (PGTSs) which incorporate probabilistic behavior into graph transformation systems. Among other applications, PGTSs permit to model randomized protocols in distributed and mobile systems, and systems with on-demand probabilistic failures, such as message losses in unreliable communication media. We define the semantics of PGTSs in terms of Markov decision processes and employ probabilistic model checking for the quantitative analysis of finite-state PGTS models. We present tool support using Henshin and Prism for the modeling and analysis and discuss a probabilistic broadcast case study for wireless sensor networks.

1 Introduction

Graph transformation systems (GTSs) provide a natural and expressive formalism for modeling dynamic distributed and mobile systems. In the recent past, extensions of graph transformation systems with quantitative properties such as real-time [1,2] and stochastic behavior [3] have been developed to increase their expressiveness further. However, many protocols used in distributed systems also employ randomization in the form of discrete probabilistic behavior to ensure liveness properties or to optimize quality of service properties without introducing a centralized authority. Probabilistic behavior is also a key ingredient for describing on-demand random failures, such as message losses in unreliable communication media. However, such discrete probabilistic decisions are not supported by any of the existing quantitative graph transformation based modeling approaches. Furthermore, since the employed models are always abstractions of the real systems, they inevitably contain nondeterminism for which no probabilistic assumption can be made. Consequently, a modeling approach is required that also permits to combine probabilistic and nondeterministic behavior.

As a case study for probabilistic and nondeterministic behavior in distributed systems, we consider a probabilistic broadcast protocol for wireless sensor networks (WSNs) described and formally analyzed in [4]. WSNs are decentralized

* Supported through a research grant by the Hasso Plattner Institute (HPI).

H. Ehrig et al.(Eds.): ICGT 2012, LNCS 7562, pp. 311–325, 2012.

and spatially distributed networks that do not rely on an existing infrastructure, such as routers or access points. To acquire or distribute information in such networks, often a simple form of a flooding protocol is employed, where *flooding* means that a node that receives a message forwards it to all its neighbors by a broadcast. However, the nodes in a WSN typically have to work with very limited resources such that unnecessary communications should be kept at a minimum in order to save energy. So-called gossiping protocols use randomization in order to reduce this overhead. In a gossiping protocol, every node decides with a certain probability whether to forward a received message or not, which reduces the communication costs. While the local decision whether to forward a received message or not requires probabilistic behavior, the asynchronous nature of the message delivery in such a network requires nondeterministic behavior.

In this paper, we introduce *probabilistic graph transformation systems* (PGTSs) which permit to describe both probabilistic and nondeterministic phenomena, and develop methods for their quantitative analysis. Transformation rules in PGTSs can have multiple right-hand sides, each of them annotated with a probability. The choice for an applicable rule and a particular match is nondeterministic, whereas the effect of a rule is probabilistic. We define the semantics of PGTSs in terms of Markov decision processes (MDPs) and employ probabilistic model checking for the quantitative analysis of finite-state PGTS models. We present tool support for the modeling and analysis of PGTSs using the Henshin [5] graph transformation tool and the probabilistic model checker Prism [6]. We discuss some of the advantages of PGTSs over component-based modeling approaches using the WSN case study presented in [4]. Briefly, PGTSs provide a better modeling scalability as the complexity of the model does not grow with the complexity of the topology. Also, models in the graph transformation-based approach can be more easily adjusted to reflect topology or protocol changes.

Organization Section 2 and 3 recall the formal foundations, specifically typed graph transformation systems, Markov decision processes and the probabilistic logic PCTL. We use the case study to illustrate their particular capabilities. In Section 4 we introduce probabilistic graph transformation systems as a modeling language and define their semantics. In Section 5, we present a probabilistic model of our case study and compare it to existing models. In Section 6 we present our tool support and discuss the obtained analysis results for the case study. Section 7 contains related work and Section 8 conclusions and future work.

2 Typed Graph Transformation

We follow the double pushout (DPO) approach for typed graph transformation [7,8], which builds on category theory. Note that our probabilistic extensions could be also applied to the single pushout (SPO) approach.

Definition 1 (Typed graphs and graph morphisms).

- *A* graph $G = \langle V, E, s, t \rangle$ *consists of a set of nodes* V*, a set of edges* E *and source and target functions* $s, t : E \to V$*.*

- *A* graph morphism $f : G_1 \to G_2$ *is a pair of functions* $f = \langle f_V, f_E \rangle$ *with* $f_V : V_1 \to V_2$, $f_E : E_1 \to E_2$, *such that* $f_V \circ s_1 = s_2 \circ f_E$, $f_V \circ t_1 = t_2 \circ f_E$.
- *Let* T *be a graph, called a* type graph. *A* typed graph $\langle G, \tau \rangle$ *consists of a graph* G *and a graph morphism* $\tau : G \to T$.
- *For two typed graphs* $\langle G_i, \tau_i \rangle$ *with* $i \in \{1, 2\}$ *over the same type graph, a* typed graph morphism *is a graph morphism* $f : G_1 \to G_2$ *with* $\tau_2 = f \circ \tau_1$.

Definition 2 (Rule). *A* rule $p = \langle L \xleftarrow{\ell} K \xrightarrow{r} R \rangle$ *is a span of injective typed graph morphisms. The graph* L *is called the left-hand side (LHS), and* R *the right-hand side (RHS) of* p.

Definition 3 (Transformation). *Given a rule* $p = \langle L \xleftarrow{\ell} K \xrightarrow{r} R \rangle$, *a typed graph* M, *and a typed graph morphism* $m : L \to G$, *called a* match. *A* transformation $M \overset{p,m}{\Longrightarrow} N$ *is defined by the double pushout diagram in Fig. 1.*

Operationally, the graph M is transformed by (1) removing the occurrence of $L \backslash \ell(K)$ in M, yielding the graph C, and (2) adding a copy of $R \backslash r(K)$ to C. A rule is applicable w.r.t. a given match, if the so-called gluing condition is satisfied. Informally, all dangling edges must be explicitly removed by the rule and all non-injectively matched nodes and edges must be consistently removed or preserved by the rule.

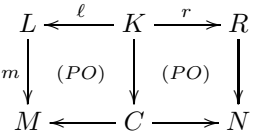


Fig. 1. DPO diagram

Negative Application Conditions. To increase the expressiveness of rules, several extensions of the basic format in Definition 2 are available. For instance, negative application conditions (NACs) provide a means to restrict the applicability of rules. Formally, a NAC is a pair $\langle N, c \rangle$ with N a typed graph and $c : L \to N$ a typed graph morphism from the rule's LHS into N. The applicability of the rule is restricted to those matches which cannot be extended to any of its NACs.

Nested Rules and Amalgamation. Nested rules provide a concept to extend a match of a basic rule to an unbounded number of substructures and to perform modifications to all these structures in an atomic step. Formally, a nested rule is modeled by a possibly nested embedding of one rule into another. The application of a nested rule can be carried out by constructing an *amalgamated* rule and applying it as a normal transformation as in Definition 3. For a comprehensive discussion of the formal foundations of parallel rule applications we refer to [9]. Regarding tooling, we use nested rules as supported by the approaches in [10,5]. For the examples in this paper, we require only nested rules of depth 1.

As our case study, we model the variant of the gossiping protocol with nondeterministic execution order and message collisions as presented in [4] using graph transformation. The type graph for this example is depicted in Fig. 2. Wireless sensors are modeled as nodes and network topologies using edges between such nodes. We usually assume bidirectional connections which we formally model

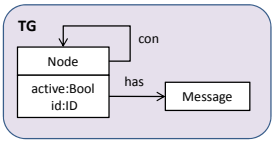


Fig. 2. Type graph

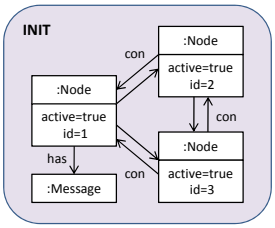


Fig. 3. Initial graph for a simple topology

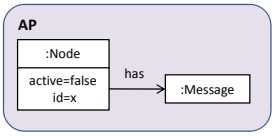


Fig. 4. Atomic proposition $received(x)$

using two edges in opposite directions. Additionally, every node can hold references to an unbounded number of messages. To identify nodes and to model their status, we use an attribute id of the finite type $\mathsf{ID} = \{1, \ldots, n\}$ and a Boolean attribute active. Note that we did not formally introduce attributes. However, attributes over finite data domains can be easily encoded in graphs.

The behavior of the gossiping protocol is modeled using three rules. Fig. 5 depicts the rule $send_1$ which models the situation where a node decides to broadcast its message to all its neighbor nodes. We depict only the LHS and the RHS and indicate the partial mapping between them using indices. A node becomes inactive if it correctly received a message. The broadcasting is possible only as long as the sender is active and has exactly one message (ensured using a NAC). If a node has more than one message, a collision occurred. Informally, if multiple neighbors send messages to the same node, it can happen that the communication is disturbed and that the node receives only noise. We use a nested rule to model the synchronous broadcast to all neighbors, i.e., every connected node receives a copy of the original message. Fig. 6 depicts the rule $send_2$ which models that the node decides *not* to broadcast the message to its neighbors. The node can become inactive only if it correctly received the message (ensured using a NAC). Fig. 7 depicts the rule *reset* which allows a node to reset itself by deleting all its messages in the case of a collision. A simple network topology consisting of only three nodes describing a possible initial graph is depicted in Fig. 3.

This model of the gossiping protocol contains only nondeterministic behavior. During the execution, multiple nodes in the network may be able to send a message at the same time. In our model, the choice for a particular sending order is nondeterministic, which allows us to capture unknown details of the network, such as the internal behavior of the nodes and the network characteristics, e.g., varying signal strengths. Note also that the nondeterministic modeling enables us to reason about the range of possible behaviors, particularly about worst-case and best-case execution orders. However, in the used approach also the fact *whether* a node forwards a message or not has to be modeled as nondeterministic, even though this decision should be probabilistic according to the gossiping protocol. In particular, it is not possible to quantitatively specify the likelihood of the message forwarding. Similarly, it is also not possible to specify probabilities for message losses due to communication in unreliable media.

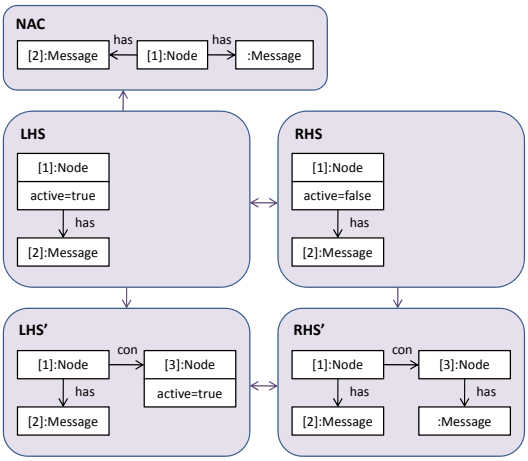


Fig. 5. Rule $send_1$

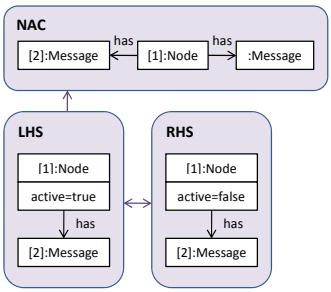


Fig. 6. Rule $send_2$

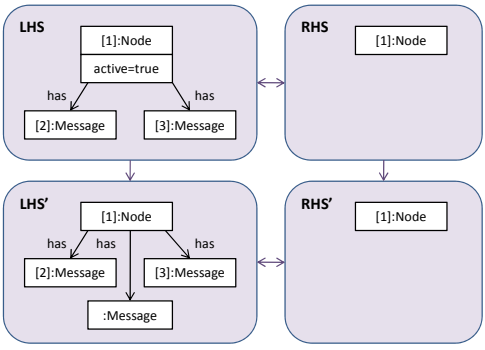


Fig. 7. Rule *reset*

Fig. 8 depicts the state space for the example with the initial configuration in Fig. 3 as a labeled transition system (LTS) in which states correspond to graphs and transitions to rule applications. For a formal definition of the derived state spaces see [11,3]. Note that both the asynchronous execution order and the local decision whether a message is forwarded is nondeterministic in this model.

To later reason about the derived state spaces, we use a specification format for graph-based atomic propositions and define a derived state labeling function. In the following, we use the notation $G \overset{p}{\Longrightarrow}$ to denote that there exists a graph G' and a match m such that $G \overset{p,m}{\Longrightarrow} G'$.

Definition 4 (Atomic propositions and labeling functions). *An* atomic proposition *is a non-modifying rule (with identical LHS and RHS). Let AP be a set of atomic propositions and Q a set of typed graphs. The* labeling function $L_Q^{AP} : Q \to 2^{AP}$ *is defined as:* $L_Q^{AP}(G) = \{\, a \in AP \mid G \overset{a}{\Longrightarrow} \,\}$ *for all* $G \in Q$.

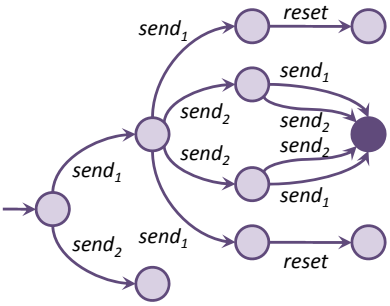


Fig. 8. Labeled transition system

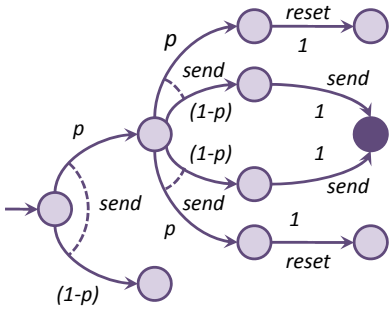


Fig. 9. Markov decision process

Thus, a graph G satisfies an atomic proposition in form of a non-modifying rule if it is applicable to G. For our example, we define the parameterized atomic proposition *received*(x) depicted in Fig. 4, where only the LHS of the non-modifying rule is shown. The parameter x ranges over the set of node IDs. Intuitively, a state satisfies *received*(x) if the node x successfully received a message and became inactive. As an example, we marked the (only) state where *received*(x) holds for all $x \in \{1, 2, 3\}$ by a filled circle in the state space depicted in Fig. 8.

3 Markov Decision Processes and Probabilistic Logic

Markov decision processes (MDPs) are a discrete-time model for systems exhibiting both probabilistic and nondeterministic behavior.

Definition 5 (Discrete probability distribution). *For a denumerable set Q, we denote with $Dist(Q)$ the set of* discrete probability distributions *over Q, i.e., the set of all functions $\mu : Q \to [0, 1]$ with $\sum_{q \in Q} \mu(q) = 1$.*

Definition 6 (Markov decision process). *A* Markov decision process *(MDP) $\mathcal{M} = (Q, q_{init}, Steps)$ consists of a denumerable set of states Q, an initial state $q_{init} \in Q$ and a probabilistic transition function $Steps : Q \to 2^{Dist(Q)}$.*

Note that *Steps* assigns a *set* of probability distributions to states in order to incorporate nondeterministic choice. Fig. 9 depicts the required MDP for the WSN example. In contrast to the LTS in Fig. 8, the local decision whether a particular node forwards a message or not, i.e., whether $send_1$ or $send_2$ is applied for a given match, is probabilistic in this model. The intuition is that the two basic rules $send_1$ and $send_2$ are combined into one probabilistic rule *send* which yields different results according to a given probability distribution. Specifically, the message is forwarded with a probability of p, and not forwarded with a probability of $(1 - p)$. However, the decision which of the enabled probability distributions is chosen remains nondeterministic. Thus, in contrast to the LTS, the MDP allows us to describe both the nondeterministic execution order of the message sending and the probabilistic decision whether to forward a message.

Formally, the operational semantics of an MDP can be understood as follows. A *probabilistic transition*, written as $q \xrightarrow{\mu} q'$, is made from a state $q \in Q$ by:

1. nondeterministically selecting a distribution $\mu \in \mathit{Steps}(q)$, and
2. making a probabilistic choice of target state q' according to μ.

A *path* of an MDP is a non-empty finite or infinite sequence of probabilistic transitions:

$$\omega = q_0 \xrightarrow{\mu_0} q_1 \xrightarrow{\mu_1} q_2 \xrightarrow{\mu_2} \ldots$$

where for all $i \in \mathbb{N}$ it holds that $q_i \in Q$, $\mu_i \in \mathit{Steps}(q_i)$, and $\mu_i(q_{i+1}) > 0$. We denote with $\omega(i)$ the ith state of ω, and with $\mathit{last}(\omega)$ the last state of ω if it is finite. An *adversary* is a particular resolution of the nondeterminism in an MDP. Formally, an adversary A for $\mathcal{M}$ is a function mapping every finite path ω of $\mathcal{M}$ to a distribution $\mu \in \mathit{Steps}(\mathit{last}(\omega))$. The set of all adversaries of $\mathcal{M}$ is denoted by $\mathit{Adv}_{\mathcal{M}}$. For any $q \in Q$ and adversary $A \in \mathit{Adv}_{\mathcal{M}}$, we let $\mathit{Paths}^A_{fin}(q)$ and $\mathit{Paths}^A(q)$ be the sets of all finite and infinite paths starting in q that correspond to A, respectively. Under a given adversary, the behavior of an MDP is purely probabilistic. Formally, an adversary for an MDP induces a probability measure Prob^A_q over the set of paths $\mathit{Paths}^A(q)$ (cf. [13] for details).

For the specification of properties of probabilistic systems, the Probabilistic Computation Tree Logic (PCTL) [14] can be used. PCTL is a branching-temporal logic based on CTL in which the existential and universal path quantifiers are replaced by a probabilistic operator which can be used to specify that the probability for a path formula meets a given lower or upper bound. Formally, the syntax of PCTL is defined as follows. A *state formula* over a set AP of atomic propositions is formed using the following grammar:

$$\Phi \quad ::= \quad \mathsf{true} \quad | \quad a \quad | \quad \neg\, \Phi \quad | \quad \Phi_1 \wedge \Phi_2 \quad | \quad \mathbb{P}_{\sim\lambda}(\phi)$$

where $a \in AP$, ϕ is a path formula, $\sim \in \{<, \leq, \geq, >\}$ and $\lambda \in [0, 1]$. A *path formula* is formed using the following grammar:

$$\phi \quad ::= \quad \bigcirc \Phi \quad | \quad \Phi_1 \,\mathsf{U}\, \Phi_2 \quad | \quad \Phi_1 \,\mathsf{U}^{\leq n}\, \Phi_2$$

where Φ, Φ_1, Φ_2 are state formulas and $n \in \mathbb{N}$. The temporal operators $\bigcirc$ and U are the next- and until-operators from CTL. $\Phi_1 \,\mathsf{U}^{\leq n}\, \Phi_2$ is a step-bounded variant of the until-operator, which states that Φ_2 holds within at most n steps, while Φ_1 holds in all states visited before a Φ_2-state was reached. The eventually-operator $\Diamond$ can be derived by setting $\Diamond\, \Phi = \mathsf{true}\, \mathsf{U}\, \Phi$ and analogously for a step-bounded variant of it. For example, using the graph-based atomic proposition in Fig. 4, the property 'with a probability of 0.95 or higher, node 2 correctly receives a message within 5 execution steps' can be formalized as $\mathbb{P}_{\geq 0.95}(\Diamond^{\leq 5} \mathit{received}(2))$.

The semantics for PCTL is defined using a satisfaction relation. Given a labeling function $L : Q \to 2^{AP}$ associating atomic propositions to states, the satisfaction relation for state formulas is defined as:

$q \models \mathsf{true}$			$q \models \Phi_1 \wedge \Phi_2$	$\Leftrightarrow$	$q \models \Phi_1$ and $q \models \Phi_2$
$q \models a$	$\Leftrightarrow$	$a \in L(q)$	$q \models \mathbb{P}_{\geq\lambda}(\phi)$	$\Leftrightarrow$	$p^{\min}_q(\phi) \geq \lambda$
$q \models \neg\, \Phi$	$\Leftrightarrow$	$q \not\models \Phi$	$q \models \mathbb{P}_{\leq\lambda}(\phi)$	$\Leftrightarrow$	$p^{\max}_q(\phi) \leq \lambda$

where $p_q^{\min}(\phi)$ and $p_q^{\max}(\phi)$ are the minimum and the maximum probabilities for the set of paths starting in q and fullfiling ϕ, formally:

$$p_q^{\min}(\phi) = \inf_{A \in Adv_{\mathcal{M}}} p_q^A(\phi) \qquad \text{and} \qquad p_q^{\max}(\phi) = \sup_{A \in Adv_{\mathcal{M}}} p_q^A(\phi)$$

where for a given adversary A and a start state q, the probability for ϕ is:

$$p_q^A(\phi) = Prob_q^A\{\omega \in Paths(q) \mid \omega \models \phi\}$$

The satisfaction relation for path formulas is defined as follows:

$\omega \models \bigcirc \Phi$	$\Leftrightarrow$	$\omega(1) \models \Phi$
$\omega \models \Phi_1 \mathsf{U} \Phi_2$	$\Leftrightarrow$	$\exists j \geq 0 : (\omega(j) \models \Phi_2 \wedge (\forall 0 \leq k < j : \omega(k) \models \Phi_1))$
$\omega \models \Phi_1 \mathsf{U}^{\leq n} \Phi_2$	$\Leftrightarrow$	$\exists 0 \leq j \leq n : (\omega(j) \models \Phi_2 \wedge (\forall 0 \leq k < j : \omega(k) \models \Phi_1))$

4 Probabilistic Graph Transformation Systems

We now introduce probabilistic graph transformations systems (PGTSs), in which the format for rules is extended to incorporate probabilistic behavior.

Definition 7 (Probabilistic rule). *A* probabilistic rule $\pi = \langle J, P, \mu \rangle$ *consists of a typed graph J, a finite, non-empty set of rules P, such that $J = L$ for all $p = \langle L \xleftarrow{\ell} K \xrightarrow{r} R \rangle \in P$, and a probability distribution $\mu \in Dist(P)$.*

A probabilistic rule $\pi = \langle J, P, \mu \rangle$ formally consists of a finite set of basic (non-probabilistic) rules P with the same left-hand side J and a probability distribution μ over these basic rules. A probabilistic rule is interpreted as a single rule with multiple right-hand sides, which are picked randomly according to μ. Basic (non-probabilistic) rules are modeled as probabilistic rules with a single RHS and a probability distribution that assigns 1 to this RHS.

Definition 8 (Probabilistic transformation). *Let M be a typed graph, $\pi = \langle J, P, \mu \rangle$ a probabilistic rule, $m : J \to M$ a match, and $p \in P$ a basic rule. A* probabilistic transformation $M \overset{\pi,m,p}{\Longrightarrow} N$ *is defined by a basic transformation $M \overset{p,m}{\Longrightarrow} N$ if and only if:*

1. *for all $p' \in P$ there exists a typed graph N' such that $M \overset{p',m}{\Longrightarrow} N'$ and*
2. *$\mu(p) > 0$.*

Thus, a probabilistic transformation is possible if and only if (1) all its basic rules are enabled, and (2) the probability for the chosen basic rule is strictly greater than zero. Note that therefore a probabilistic rule is applicable w.r.t. a match only if the gluing condition is satisfied for all its basic rules. This is necessary because the choice for a particular basic rule is random and thus it must be ensured that all of them are enabled. Note, however, in the case of SPO graph transformation semantics, no checking of the gluing condition is required.

Definition 9 (Probabilistic graph transformation system). *A* probabilistic graph transformation system *(PGTS) is a tuple* $\mathcal{G} = \langle T, G_{init}, \Pi \rangle$ *consisting of a type graph* T*, an initial graph* G_{init} *typed over* T*, and a set of probabilistic rules* Π *typed over* T*.*

In a given PGTS, a probabilistic transformation $M \overset{\pi,m,p}{\Longrightarrow} N$ is made by:

1. nondeterministically selecting an applicable rule $\pi = \langle J, P, \mu \rangle \in \Pi$,
2. nondeterministically selecting a match $m : J \to M$,
3. making a probabilistic choice for a basic rule $p \in P$ according to μ,
4. transforming M into N using the basic rule p and the match m.

Thus, a probabilistic transformation $M \overset{\pi,m,p}{\Longrightarrow} N$ is a particular resolution of both the nondeterministic and the probabilistic choices in a PGTS. We denote with $G_0 \Longrightarrow^*_{\mathcal{G}} G_n$ the fact that there exists a finite sequence of consecutive probabilistic transformations using the probabilistic rules of $\mathcal{G}$:

$$G_0 \xRightarrow{\pi_0,m_0,p_0} G_1 \xRightarrow{\pi_1,m_1,p_1} \ldots \xRightarrow{\pi_{(n-1)},m_{(n-1)},p_{(n-1)}} G_n$$

The operational semantics of a PGTS induces a Markov decision process.

Proposition 1 (Induced MDP). *Let* $\mathcal{G} = \langle T, G_{init}, \Pi \rangle$ *be a PGTS. Then* $\mathcal{G}$ *induces a Markov decision process* $\mathcal{M}_{\mathcal{G}} = \langle Q, q_{init}, Steps \rangle$ *with:*

- $Q = \{[G] \mid G_{init} \Longrightarrow^*_{\mathcal{G}} G\}$*, i.e., the set of isomorphism classes of typed graphs reachable from* G_{init}*,*
- $q_{init} = [G_{init}]$*,*
- $Steps([G]) = \{\ \nu \mid G \overset{\pi,m}{\Longrightarrow} \nu\ \}$ *where* $G \overset{\pi,m}{\Longrightarrow} \nu$ *with* $\pi = \langle J, P, \mu \rangle \in \Pi$ *denotes the fact that there exists* $p \in P$ *and* $G' \in Q$ *such that* $G \overset{\pi,m,p}{\Longrightarrow} G'$*, and where* $\nu \in Dist(Q)$ *is induced by* μ *as follows:*[1]

$$\nu([G']) = \sum_{p \in P : G \overset{\pi,m,p}{\Longrightarrow} G'} \mu(p) \tag{1}$$

The induced probabilistic transitions are defined in (1) by associating the probabilities of each basic rule p to the result of applying p to the current graph with the chosen match. Note that the states are defined up to graph isomorphism and that the sum in (1) is required for cases with identical or symmetric RHSs.

The concepts of graph-based atomic propositions, negative application conditions and nested rules as described in Section 2 can be directly transferred to PGTSs. NACs are defined in the usual way and restrict the applicability of a probabilistic rule as a whole. Nesting of probabilistic rules is achieved by a nesting of its basic rules, where the nested LHSs of all basic rules must be identical. For simplicity, we restrict ourselves to nested rules of depth 1 here. Note that the probabilities are associated only to the embedded rule (which is matched only once) in a nested rule. Due to lack of space, we omit the formal definition here and illustrate the concepts using an example.

[1] We use the convention $\sum_{\emptyset} = 0$ for sums over empty sets.

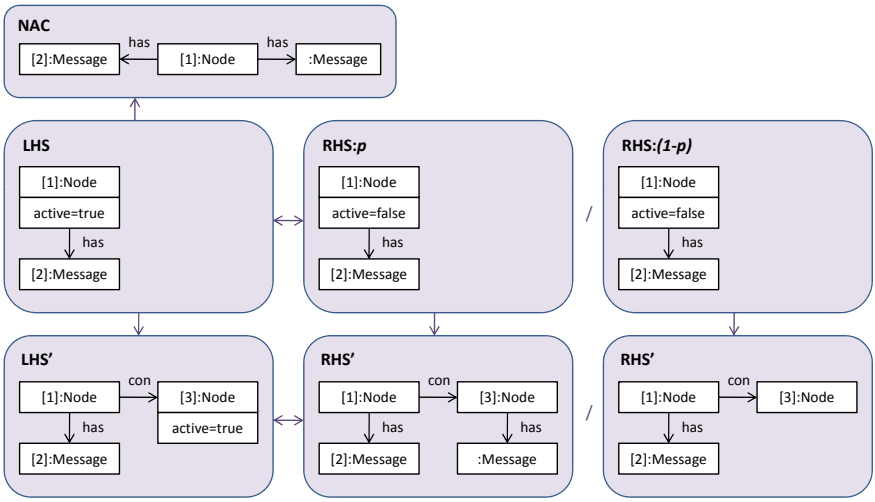


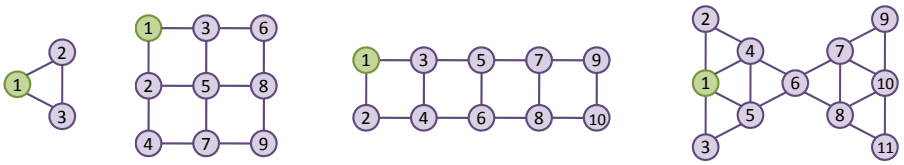
Fig. 10. Probabilistic rule *send*

5 Modeling and Comparison

In this section, we show how to use probabilistic graph transformation systems for a faithful modeling of the gossiping protocol described in Section 1 and 2 and discuss the differences to component-based models. For the PGTS model we reuse the type graph in Fig. 2, the initial graph in Fig. 3 and the rule *reset* in Fig. 7, where we trivially associate a probability of 1 to its only RHS. To specify the likelihood of the message forwarding, we combine the two basic rules $send_1$ and $send_2$ in Fig. 5 and 6 into one probabilistic rule *send* with two RHSs, depicted in Fig. 10. The first RHS models the case where the message is forwarded with a probability of p, whereas the second RHS models the case where the message is not forwarded with a probability of $(1 - p)$. Note that in both cases the node becomes inactive and that the probabilistic rule is enabled only if no collision occured. Moreover, the synchronous message passing to all neighbors is modeled again using a nested rule. To reason about this model, we reuse the atomic proposition $received(x)$ in Fig. 4. As initial graphs, we consider four example topologies shown in Fig. 11, where in each network the broadcasting starts at node 1. Network 11a) is formally modeled by the typed graph in Fig. 3.

In the following, we discuss some of the advantages of using PGTSs as a modeling approach over traditional component-based modeling approaches as employed, e.g., in [4], where automata or process algebra models are used to define the behavior of the components and the system as a whole.

Modeling scalability. The first important observation is that the size of the topology has only a minor impact on the size of the PGTS model. Switching from the simple topology depicted in Fig. 11a) to the 3×3 network in Fig. 11b) only required to exchange the initial topology while the rules and the type graph remain



(a) (b) (c) (d)

Fig. 11. Schematic example network topologies (broadcasting starts at node 1)

the same. In contrast, when using component-based models such as employed in [4], a larger network topology requires that additional components, each with its own specific local behavior and communication with its neighbors must be added to the specification. Scalability of the verification approach, however, is a separate issue and is planned to be addressed in our future work.

Changeability. When modeling different network topologies with PGTSs, this simply boils down to using different input graphs as initial states such as the four different example topologies in Fig. 11. In contrast, a modification in the network topology is a real challenge when using component-based models, since the intuitive graph structure of the network is not tangible in the specification and must be carefully encoded in the local behavior of the nodes. Moreover, in the case of a change in the protocol, only a few rules in the PGTS need to be adjusted, whereas in an component-based model the local specifications of all nodes must be altered to reflect the change in the protocol.

Expressiveness. In [4], multiple versions of the gossiping protocol are considered, which can be all modeled using PGTSs. The model in this paper corresponds to the case with nondeterministic execution order. The synchronous versions can be modeled as a PGTS by increasing the nesting depth of the rule *send* such that all active nodes with exactly one message execute the probabilistic sending at the same time. The last variant presented in [4] includes a simple, i.e., memoryless probabilistic delay for the sending of messages. This can be modeled also in a PGTS by adding a Boolean attribute which is used as an additional precondition for the *send* rule and which is switched on by a probabilistic rule.

Moreover, we argue that modeling dynamic structural changes in the network topology is (except for encodings of very simple cases) impossible using component-based models. In contrast, in the graph transformation-based approach, dynamic structural changes as needed for modeling reconfigurable and mobile systems can be expressed directly.

Simplicity. We believe that specifying the distributed protocol using a PGTS is less intricate and less error-prone because there is a clear separation between the description of the protocol modeled using the rules on the one hand, and the particular network structure on the other. This hypothesis is to some extent supported by our modeling experiment. In particular, we compared our results discussed in Section 6 for network (b) to the data presented in [4]. While

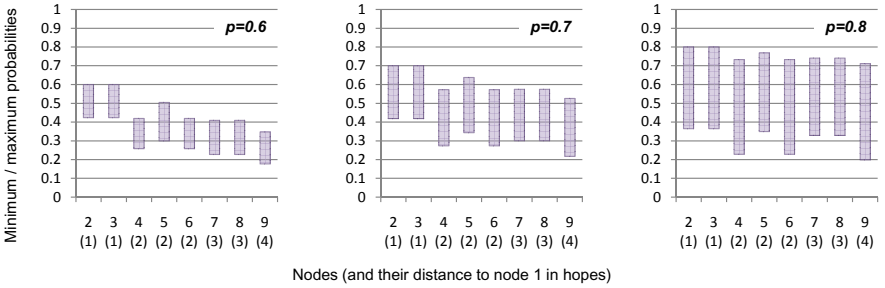


Fig. 12. Minimum / maximum probabilities for message reception for each node in network (b) and send probabilities of $p = 0.6, 0.7$ and 0.8

our experiments yielded the same maximum probabilities, we noticed that our modeling apparently predicted smaller minimum probabilities for the message receptions. Apparently, their specification contains a modeling error as the probabilistic decision whether to forward a message or not is already done at the message reception, which rules out the possibility of a collision in the case the node decides not to forward the message. However, due to the complicated encoding of the protocol this difference between their specification and the MDP induced by our model could be identified only by a detailed analysis.

6 Tool Support and Analysis

We have implemented tool support for PGTSs in version 0.9.2 of the Henshin [5] graph transformation tool using Prism 4 [6] as probabilistic model checking back-end. We model probabilistic rules in Henshin using multiple basic graph transformation rules with the same LHS (and possible additional application conditions). Probabilities are associated to the different basic rules using annotations. We then use Henshin's state space generation capabilities to derive an LTS, which is subsequently converted into an MDP by (1) removing all illegal transitions where not all basic rules of a probabilistic rule are applicable for the same match, and (2) replacing the nondeterministic choice between annotated basic rules by probabilistic transitions. Our extension of Henshin generates an MDP in the input format of Prism to carry out the PCTL model checking and for computing the minimum and the maximum probabilities.

Using our tool, we have modeled the gossiping protocol and ran a number of experiments. As a first setting, we fixed the send probability to $p = 0.8$ and chose the network topology (b). For these parameters, we verified using Prism that the property $\mathbb{P}_{>0.3}(\Diamond\ \mathit{received}(8))$ holds, i.e., the probability that node 8 receives the message is greater than 0.3. In addition to the checking of PCTL formulas, we used Prism to compute the minimum and the maximum probabilities for each node successfully receiving the message. Fig. 12 depicts the minimum and the maximum probabilities for each node in network (b) correctly receiving the

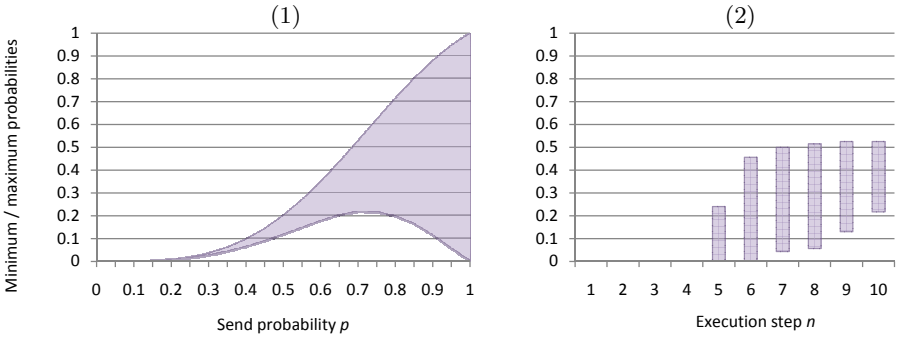


Fig. 13. Min./max. probabilities for message reception of node 9 in network (b)

message and for send probabilities of $p = 0.6, 0.7$ and 0.8. The minimum probabilities reflect worst-case, the maximum probabilities best-case execution orders for each node. We note that the minimum (and the maximum) probabilities for the different nodes vary more or less depending on the chosen send probability and the location of the node in the grid. To illustrate the impact of the send probability, we have plotted the minimum and the maximum reception probabilities for node 9 with changing p in Fig. 13.1). Note that for values of p greater than approx. 0.7, the minimum reception probability decreases again.

We further investigated how the probability for a specific node receiving the message changes over time, where time is measured as discrete execution steps. Such properties can be specified using the step-bounded until-operator in PCTL. Specifically, fixing the send probability to $p = 0.7$, we verified that the property $\mathbb{P}_{\geq 0.2}(\Diamond^{\leq 10}\ \mathit{received}(9))$ holds, i.e., the probability that node 9 in network (b) successfully received the message after 10 execution steps is at least 0.2. Additinally, Fig. 13.2) depicts the minimum and the maximum probabilities for node 9 having received the message after 1..10 execution steps.

Due to the graph-based approach, models with different network topologies can be easily derived. The minimum and maximum probabilities for the networks (b)-(d) are depicted in Fig. 14. The probabilities drop more for the nodes in network (c) with high indizes than in network (b) which is caused by the higher distance and the fewer number of connections. For network (d), the differences between the minimum and maximum probabilities are higher than in the other networks. This is caused by the higher connectivity of the network which increases the chance for collisions. It is also evident that node 6 is a bottleneck in the network causing the probabilities to drop abruptly for nodes 7-11.

7 Related Work

As discussed in Section 5, PGTSs compared to component-based models (as, e.g., in the PRISM specification language), provide a greater expressiveness in terms of modeling concepts, since there is a clear separation between the modeled protocols on the one hand and the used network topologies on the other.

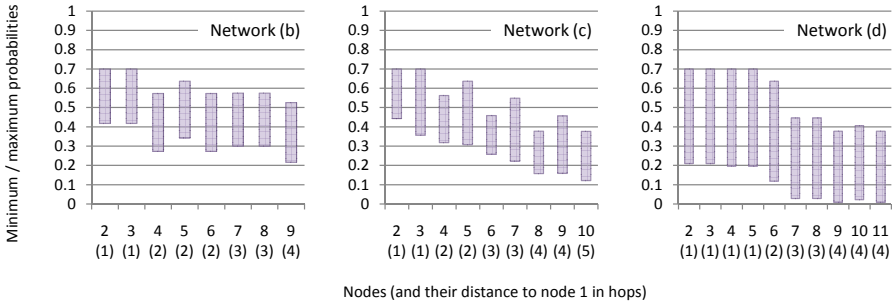


Fig. 14. Minimum / maximum probabilities for message reception for networks (b)-(d) with a fixed send probability of $p = 0.7$

Therefore, PGTSs permit to study different network topologies and to adjust protocols with minimal effort. In contrast, the component-based approaches require to encode the topology into the local behavior of the nodes, which can result in complex and erroneous specifications.

Executable term rewrite theories as used in Maude [15] provide similarly to GTSs natural modeling concepts for concurrent systems with structure dynamics. Probabilistic rewrite theories in PMaude [16] provide a combination of structure dynamics, probabilistic behavior for discrete branching, and stochastic behavior. Properties for such models can be specified using probabilistic temporal logics and checked using discrete event simulation. However, in order to simulate and analyze models in PMaude, all nondeterminism has to be resolved, i.e., nondeterministic choices for rules and matches as in PGTSs are not allowed.

Several extensions of GTSs with quantitative properties such as real-time [1,2] and stochastic behavior [3] exist. However, the combination of discrete probabilistic decisions and nondeterminism in PGTSs can be emulated neither by real-time nor by stochastic models. To clarify this, we discuss in detail the difference to stochastic graph transformation systems (SGTSs) [3]. While SGTSs are based on a continuous time model, PGTSs are based on a discrete one. Furthermore, SGTSs do not support nondeterminism. Instead, in any given state there is a competition between all enabled rules and their matches, which is also referred to as a *race condition*. The choice for a particular rule and match is decided probabilistically based on the rules' stochastic rates. In contrast, the choice for a particular rule and match in a PGTS is made nondeterministically whereas the effect of a rule is probabilistic. Due to the different time model and the nondeterminism, PGTSs cannot be encoded into SGTSs, nor vice versa.

8 Conclusions and Future Work

In this paper, we introduced probabilistic graph transformation systems (PGTSs), provided a sound foundation based on Markov decision processes, and presented related tool support. We further demonstrated that the modeling using PGTSs

compared to existing component-based approaches as, e.g., in PRISM scale better and can be more easily adjusted to reflect changes in the topology or protocol. For future work, we plan to develop a compact visual syntax for PGTSs, to incorporate interval-valued probabilistic and real-time behavior, and to improve the scalability of the verification procedure using compositional schemes.

References

1. Gyapay, S., Varró, D., Heckel, R.: Graph transformation with time. Fundamenta Informaticae 58, 1–22 (2003)
2. Giese, H.: Modeling and Verification of Cooperative Self-adaptive Mechatronic Systems. In: Kordon, F., Sztipanovits, J. (eds.) Monterey Workshop 2005. LNCS, vol. 4322, pp. 258–280. Springer, Heidelberg (2007)
3. Heckel, R., Lajios, G., Menge, S.: Stochastic graph transformation systems. Fundamenta Informaticae 74, 63–84 (2006), doi: 1231199.1231203
4. Fehnker, A., Gao, P.: Formal Verification and Simulation for Performance Analysis for Probabilistic Broadcast Protocols. In: Kunz, T., Ravi, S.S. (eds.) ADHOC-NOW 2006. LNCS, vol. 4104, pp. 128–141. Springer, Heidelberg (2006)
5. Arendt, T., Biermann, E., Jurack, S., Krause, C., Taentzer, G.: Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part I. LNCS, vol. 6394, pp. 121–135. Springer, Heidelberg (2010)
6. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
7. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M.: Algebraic approaches to graph transformation I: Basic concepts and double pushout approach. In: Handbook of Graph Grammars and Computing by Graph Transformation, pp. 163–245. World Scientific (1997)
8. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation (Monographs in Theoretical Computer Science). Springer (2006)
9. Taentzer, G.: Parallel and Distributed Graph Transformation: Formal Description and Application to Communication-Based Systems. PhD thesis, TU Berlin (1996)
10. Rensink, A., Kuperus, J.H.: Repotting the geraniums: On nested graph transformation rules. In: GT-VMT 2009. ECEASST, vol. 18 (2009)
11. Kastenberg, H., Rensink, A.: Model Checking Dynamic States in GROOVE. In: Valmari, A. (ed.) SPIN 2006. LNCS, vol. 3925, pp. 299–305. Springer, Heidelberg (2006)
12. Kwiatkowska, M., Norman, G., Parker, D.: Game-based abstraction for Markov decision processes. In: QEST 2006, pp. 157–166. IEEE Computer Society (2006), doi: 10.1109/QEST.2006.19
13. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains, 2nd edn. Springer (1976)
14. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6, 512–535 (1994), doi: 10.1007/BF01211866
15. Clavel, M., Duran, F., Eker, S., Lincoln, P., Marti-Oliet, N., Meseguer, J., Quesada, J.: Maude: specification and programming in rewriting logic. Theoretical Computer Science 285(2), 187–243 (2002), doi: 10.1016/S0304-3975(01)00359-0
16. Agha, G., Meseguer, J., Sen, K.: PMaude: Rewrite-based specification language for probabilistic object systems. Electron. Notes Theor. Comput. Sci. 153, 213–239 (2006), doi: 10.1016/j.entcs.2005.10.040