GSPNs Revisited: Simple Semantics and New Analysis Algorithms

Joost-Pieter Katoen
Software Modelling and Verification Group, RWTH Aachen University, Germany, and
Formal Methods and Tools Group, University of Twente, The Netherlands
Email: katoen@cs.rwth-aachen.de, katoen@cs.utwente.nl

***Abstract*—This paper considers interactive Markov chains (IMCs), a natural generalization of transition systems and continuous-time Markov chains (CTMCs). We show how they can be used to provide a truly simple semantics of Generalized Stochastic Petri Nets (GSPNs). In fact, any GSPN. In particular, no restrictions are imposed on the concurrent/conflicting enabledness of immediate transitions. This contrasts with classical solutions for GSPNs which use weights. (A simple extension of IMCs also covers weights.) In addition, we will present novel analysis algorithms for expected time and long-run average time objectives of IMCs, i.e., GSPNs. Two case studies indicate the feasibility of these analyses and show that a classical reliability analysis for confused GSPNs may lead to significant over-estimations of the true probabilities. The key message is: nondeterminism is not a threat, treat it *as is*! This yields both a simple GSPN semantics and trustworthy analysis results.**

I. Performance evaluation with GSPNs

A. GSPNs in a nutshell

Generalized Stochastic Petri Nets (GSPNs [17]) are a well-established modelling formalism for performance and dependability evaluation supported by tools such as GreatSPN [9]. GSPNs support *timed* and *immediate* transitions. The former have a random duration governed by a negative exponential distribution. Immediate transitions take zero time units. GSPNs adopt the token game of classical Petri nets, except that the resolution between concurrently enabled timed transitions is probabilistic. It is resolved according to a so-called *race* policy. If two timed transitions with rate $\lambda \in \mathbb{R}_{>0}$ and $\mu \in \mathbb{R}_{>0}$ are concurrently enabled, the former wins the race with likelihood $\frac{\lambda}{\lambda+\mu}$, the second with probability $\frac{\mu}{\lambda+\mu}$. This race policy is adopted from continuous-time Markov chains (CTMCs) and generalizes in a simple way to multiple enabled timed transitions. In case a timed and an immediate transition are concurrently enabled, the latter will occur as the probability that a timed transition occurs immediately is zero. This conforms to the *maximum progress* policy: immediate transitions have priority over timed ones. The choice between concurrently enabled immediate transitions is resolved probabilistically using *weights* (possibly enriched with some priority scheme). For immediate transitions with weights $w \in \mathbb{N}_{>0}$ and $u \in \mathbb{N}_{>0}$, say, the former is selected with probability $\frac{w}{u+w}$ and the other with the remaining probability.

This research has been funded by the EU FP7 Project MoVeS and the DFG-NWO bilateral project ROCKS.

Performance evaluation of a safe GSPN –which corresponds basically to nets with a finite marking graph– proceeds at the level of its reachability graph. That graph is reduced to a finite CTMC, for which efficient steady-state and transient solvers are at hand, or that may be subject to stochastic model checking [1], [6]. Due to their formality, visual representation, and the availability of efficient evaluation support, GSPNs [1] have become popular in both academia and industry. GSPNs are also attractive as semantic model for (quantitative versions of) high-level design notations, such as extended fault trees, logistic networks, AADL, fragments of the UML, BPEL4WS. Providing a translational mapping onto GSPNs is a proper way to unambiguously define the semantics, and to get access to powerful performance evaluation tools.

B. Why revisit GSPNs?

So why should one revisit GSPNs after all? There is just a single reason: *nondeterminism*. Nondeterminism is a well-studied phenomenon in net theory. In its simplest form it occurs as a conflict between two (unweighted) immediate transitions. More involved scenarios occur in the presence of confusion. Confusion basically arises if two transitions are enabled (in some marking) and the occurrence of one of the transitions changes the set of transitions that are in conflict with the other transition, the so-called conflict set. Already in the early days of GSPNs, the problem with nondeterminism has been widely recognized: *"the solution of a conflict due to the firing order can cause very subtle problems in the stochastic definition of GSPNs"* [17, pp. 46]. The main point is that in presence of nondeterminism, the marking graph of a GSPN is not a stochastic process. This means that such GSPNs are not amenable to standard analysis of CTMCs. In this case, one can no longer speak about, e.g., *the* reachability probability of a given marking M, or *the* transient probability of M at some time $t \in \mathbb{R}_{\geq 0}$, or *the* probability to be in M on the long run. These probabilities instead depend on the resolution of the nondeterminism. Consider a simple example, cf. Fig. 1. This net has a (simple form of) confusion. In the initial marking, transitions t_0 and t_2 are enabled. Firing t_0 yields a marking in which either t_1 or t_2 can be taken, but not both. The choice between t_1 and t_2 is nondeterministic. The likelihood of reaching a marking containing p_6 is zero

[1] And related net formalisms such as stochastic activity networks (SANs).

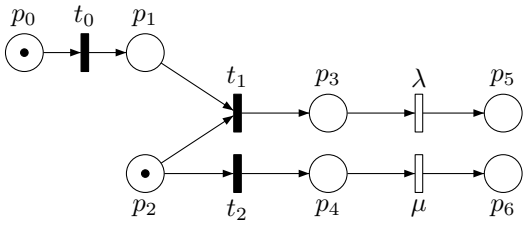

Fig. 1. A simple confused GSPN [17]. Immediate transitions are drawn as solid bars, and timed transitions as rectangles.

if after the firing of t_0, the conflict between t_1 and t_2 is resolved in favour of t_1; otherwise it is one. Associating the weight $w_i = w$ to immediate transition t_i yields that such marking is obtained with probability $\frac{3}{4}$: either t_2 fires before t_0 (which happens with probability $\frac{1}{2}$) or t_0 and t_2 fire in sequence (whose likelihood is $\frac{1}{4}$). In fact, the weight of t_0 has a major influence on the reachability probabilities, as can be seen from the general expression to eventually reach a marking containing p_6:

$$Pr\{\, \Diamond p_6 \,\} \;=\; \frac{w_2}{w_0{+}w_2} + \frac{w_0}{w_0{+}w_2}\cdot\frac{w_1}{w_1{+}w_2}.$$

However, in the initial marking t_0 is not in conflict, but concurrent with t_2. That is to say, the concurrency between t_0 and t_2 is resolved probabilistically.

C. How have GSPNs coped with confusion so far?

A simple solution is to only consider GSPNs that are confusion-free, akin to free-choice Petri nets. This is however far too restrictive. In fact, early approaches require that *"it [i.e., a GSPN] must be confusion-free at priority levels greater than zero (i.e., in subnets of immediate transitions)"* [17, pp. 110]. Another, somewhat naive, solution is to inspect the marking graph and resolve all nondeterminism in some probabilistic manner, for instance, uniformly. This however yields analysis results that are incomprehensible to the modeller and requires the generation and inspection of the marking graph. In addition, concurrency is resolved (as in the above example) probabilistically. The common approach therefore has been to develop modelling means and analysis checks at the net level. Modelling means are to equip immediate transitions with global priority levels and globally assigned weights. This allows the modeller to diminish or sometimes even delete the occurrence of nondeterminism. In the above example such weight assignment indeed yields a stochastic process. Using weights (and priorities) boils down to conflict resolution being performed probabilistically rather than using nondeterminism. However, priorities and weights do not, and cannot, eliminate nondeterminism in all cases, e.g., not for all kinds of confusion. This is not surprising, since confusion is itself a semantic phenomenon. Despite several attempts to define necessary conditions at the net level, e.g., [7], [8], [16] –extended conflict sets and refinements thereof– that ensure absence of confusion, in general the reachability graph is needed so as to check the absence of confusion. (Similar "well-specified" checks have been proposed for related formalisms such as SANs [19].) The fact that it is difficult to come up with net-level criteria is witnessed by [21] that shows the insufficiency of extended conflict sets. In fact, [21] proposes a net-level technique that is proven to yield a stochastic process as marking graph but yields spurious warnings, i.e., it is too conservative. Note that translational GSPN mappings of design notations such as the UML, AADL, and fault trees should contain an obligatory proof that the semantics is confusion-free by construction. To our knowledge these proofs are seldomly given, if at all.

D. Our approach

Nondeterminism in GSPNs is thus a well-established and well-studied problem. This is not new. Our treatment of it however will be different. The key issue is: we keep nondeterminism *as is*. This allows the treatment of a substantially larger class of GSPNs. In particular, no restrictions are imposed on the presence of confusion. First and foremost, this allows for a substantially larger class of nets. There is no burden on the modeller to ensure that certain net-level criteria are met. If needed or convenient, weights can be used to diminish the presence of nondeterminism in the net. The use of weights (and priorities) can however be restricted to cases in which the modeller has sufficient information and wants to quantify conflict resolution. Our approach is backward compatible: in case nondeterminism is absent, the same stochastic process as for classical GSPNs results. However, if nondeterminism remains –or is deliberately left present– this does not impose a restriction. Neither semantically nor from an analysis point of view. There is no need for imposing net-level constraints and no proof obligation when using GSPNs as semantic model. The resulting GSPN semantics is truly simple. The marking graph of a GSPN is a stochastic *decision* process, a.k.a. a (simple) continuous-time stochastic game. The quantitative analysis of these games yields upper and lower bounds. One thus obtains, e.g., the minimal and maximal reachability probability of a given marking, or the minimal and maximal probability to be in a certain marking at time t, and so on.

II. Semantics of GSPNs: Revisited

We suggest to use Markov automata (MA) [11], a combination of interactive Markov chains (IMCs) [14] and probabilistic automata [20], as semantic model for (safe) GSPNs. MA are finite transition systems with action and timed transitions. The latter are labelled with a rate of an exponential distribution, indicated as λ and μ before, and conenct a source state to a target state. An action transition $s \xrightarrow{a} \mu$ asserts that starting from state s the system can engage in action a and yields a probability distribution μ over the states. That is to say, state s' results with probability $\mu(s')$. If $s \xrightarrow{a} \mu$ and $s \xrightarrow{a} \nu$, a nondeterministic choice between the distributions μ and ν is made. Action transitions are subject to the *maximal progress assumption*. This means that action transitions trigger instantaneously. They thus take precedence over all timed transitions. The MA semantics of GSPNs is now rather simple. States correspond to markings. Timed transitions match one to

one. Consider the confused GSPN in Fig. 2. Fig. 3 shows

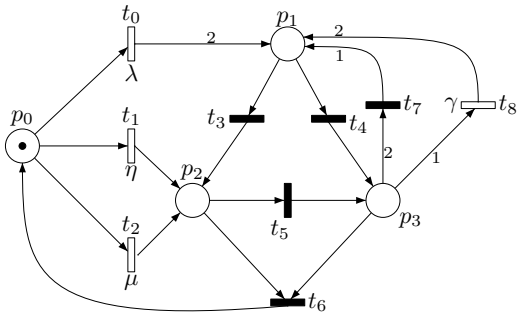

Fig. 2. An example confused (unweighted) GSPN

the marking graph of this GSPN as a Markov automaton As the GSPN does not contain weights, the action transition relation is relating a source state to a single target state (i.e., μ is a Dirac distribution). The timed transition relation is indicated by solid arrows and the immediate transition relation by dashed arrows. Tangible markings are shaded gray whereas vanishing markings (i.e., markings in which only immediate transitions are enabled) are white. Evidently, conflicts arise between transitions t_3 and t_4 (in various markings) and t_5 and t_6 (in marking 0011). All classical approaches to GSPNs

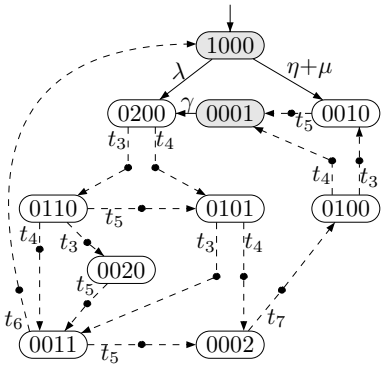

Fig. 3. The marking graph of the confused GSPN of Fig. 2

abandon the above example net, as it does not yield a CTMC and thus cannot be analysed. According to [21, Sec. 2.4], it is not well-defined. This can be seen as follows. In marking 0011, the set of reachable tangible markings is $\{\,1000, 0001\,\}$. If the enabled transition t_5 is chosen, the tangible marking 0001 is reached almost surely. However, if enabled transition t_6 is chosen, we enter the tangible marking t_8.

It remains to explain the treatment of weights. Weighted immediate transitions are represented by action transitions in the following way. Suppose in marking M, a finite number of immediate transitions t_i with weight w_i are enabled, where firing t_i yields marking M_i. Then this yields the action transition $M \xrightarrow{\tau} \mu$ where τ is some internal action, and $\mu(M_i) = \frac{w_i}{W}$ with $W = \sum w_i$. (If multiple immediate transitions yield the same marking, probabilities need to be added.)

A few remarks are in order. For confusion-free GSPNs, the action transition relation is deterministic. The resulting MA is then a CTMC up to weak bisimulation [11]. In fact, for confusion-free nets, this CTMC is isomorphic to the reachable marking graph obtained using the traditional GSPN semantics after removal of vanishing states. Secondly, MA are closed under parallel composition by synchronizing on action transitions in a CSP-like manner. They are thus compositional. This allows for building GSPNs from smaller ones in a component-wise manner. As weak bisimulation is a congruence with respect to parallel composition, it means that component-GSPNs can be treated in isolation before composing them with other component nets. Thirdly, the MA semantics covers GSPNs that may contain reachable cycles of immediate transitions. To our knowledge, existing semantics of GSPNs have not covered this so far. Finally, as MA support exponential distributions and nondeterminism, they are an appropriate candidate model for a variant of the Petri box calculus with GSPNs as denotational model. It would be interesting to work this out in detail and compare this to the Petri box calculus for GSPNs [15].

III. Quantitative Analysis of GSPNs: Revisited

Our proposed GSPN semantics is not very deep and perhaps not surprising. It is basically the usual GSPN semantics except that if nondeterminism remains, this semantically does not pose a restriction. The quest remains on how to analyze these models. State-of-the-art software tools that support the generation of interactive Markov chains, MA without discrete probabilistic branching, from high-level descriptions such as LOTOS [10], dynamic fault trees [3], or AADL [4] typically can only perform a performance analysis if the IMC after weak bisimulation minimization yields a CTMC. Then standard CTMC analysis or model-checking techniques can be used. This is however a severe restriction as in many cases nondeterminism remains after weak bisimulation. Due to recent developments in the analysis of real-time stochastic games –in fact, IMCs and MA are $1\frac{1}{2}$-player real-time stochastic games– several quantitative objectives can be analysed. In the following we consider three such objectives: expected time, long run, and time-bounded reachability objectives. It is worthwhile to mention that all these three measures are preserved by bisimulation. Thus, bisimulation-equivalent marking graphs have identical measures. In our analysis described below, the nondeterminism is resolved by policies, oracles say, that in any marking with several enabled immediate transitions select one of them. One way of resolving the nondeterminism by a policy is to pick an enabled transition in a randomized manner, just as weights at the net level. The main difference, however, is that our analysis provides results that are valid for an entire *class* of policies, whereas the analysis of a weighted GSPN would only be valid for that particular weighted resolution of the nondeterminism, i.e. for a *single* policy.

A. Expected time objectives

What is an expected time objective? Let M be a marking, and G a set of (goal) markings. The random variable T_G is the elapsed time before visiting some marking in G for the first time when starting from M. The value of T_G depends on how the nondeterminism is resolved by the policy at hand. In

fact, we are not interested in the expected value of T_G for a given policy, but in its minimal expected time for all possible policies. Stated differently, we are interested in the expected time to reach G under the worst possible policy, i.e., a policy that attempts to keep the GSPN away from all markings in G. In this setting, the most demonic policy determines the minimal expected value of T_G Dually, we are interested in the maximal expected time to reach G. This corresponds to the most angelic policy, i.e., a policy that maximally assists the GSPN to get to a marking in G. It turns out that *memoryless* policies are optimal, i.e., there is a memoryless policy that achievs the minimal (or maximal) expected time to reach G. Such policies select an enabled immediate transition in a given vanishing marking in a fully deterministic way. That is to say, such policy is just a function that associates an immediate transition to a given vanishing marking.

Evidently, for a CTMC (confusion-free GSPN), the minimal and maximal values of T_G coincide. In fact, computing expected time objectives for CTMCs boils down to solving a linear equation system. It recently has been shown that the computation of minimal (or maximal) expected time objectives in IMCs can be reduced to to a non-negative stochastic shortest path (SSP) problem in Markov decision processes (MDPs). It is well-known that such SSP problems can be solved by a linear programming problem [2] for which efficient algorithms and tools (such as SOPLEX) exist. Details of this reduction and the algorithms can be found in [12]. First experiments indicate that computing expected time objectives scales rather well. Our current investigations show that the reduction to SSP problems can easily be adapted to MA.

B. Long run objectives

As a next performance measure, we consider long-run objectives. As before, let M be a marking and G a set of markings. Let $\mathbf{I}_G$ be an indicator function with $\mathbf{I}_G(M) = 1$ if $M \in G$ and 0, otherwise. The fraction of time spent in G on an infinite path π up to time bound $t \in \mathbb{R}_{\geq 0}$ is given by the random variable

$$A_{G,t}(\pi) \;=\; \frac{1}{t}\int_0^t \mathbf{I}_G(\pi@u)\; du$$

where $\pi@t$ denotes the marking along path π at time point t. A path through an MA is an alternating sequence of markings and either an action or a non-negative delay. Recall that G only contains tangible markings, such that $\pi@t$ is a single marking. Let A_G be the random variable obtained when taking the limit of t to infinity of $A_{G,t}$. We are now interested in the expected value of this random variable. As for expected time objectives, we do not focus on a single given policy, but on the expected minimal and maximal time spent in some marking in G on the long run. As for expected time objectives, it can be shown that memoryless policies achieve the minimal (or maximal) long run time to spent in G. For all CTMCs (i.e., confusion-free GSPNs), the expected minimal and maximal long-run time is equal. In case the CTMC is ergodic, this measure coincides with the stationary probability to be in G. (Stationary probabilities are one of the most used measures for GSPN analysis.) We may assume w.l.o.g. that G only contains tangible markings, as the long-run average time spent in any vanishing marking state is always zero. This claim follows directly from the fact that states that only have outgoing action transitions are instantaneous, i.e., their sojourn time is 0 by definition. The general idea of computing the minimal long-run time spent in G is the following three-step procedure:

1) Determine the maximal end components $\{L_1, \ldots, L_k\}$ of the MA at hand.
2) Determine the minimal long-run time spent in G within each maximal end component L_j.
3) Solve a stochastic shortest path problem.

The first step is performed by a graph-based algorithm, whereas the last two steps boil down to solving linear programming problems. Determining the minimal expected long-run time in an end component can be reduced to a long-run ratio objective in an MDP equipped with two cost functions. For details, algorithms and proofs we refer to [12].

C. Time-bounded reachability objectives

Let M be a marking and G a set of tangible markings. We are interested in the minimal (or maximal) probability to reach some marking G from M within a given deadline $d \in \mathbb{R}_{\geq 0}$. This problem is much harder than the previous two. One of the main reasons is that memoryless policies are not optimal. In fact a policy that selects on the basis of the current state and the total elapsed so far, is optimal. Intuitively speaking, this is understandable. Consider a state in which there is a choice between almost surely but slowly reaching G, or reaching G quickly with the risk of not reaching G at all. If there is just a little time left until the deadline d is reached, it is optimal to gamble and choose the fast alternative as it is more likely to reach G soon than choosing the safe but slow alternative. In case there is sufficient time left, an optimal policy rather plays safe and chooses the slow alternative. The point is that the total elapsed time so far is a real, continuous value. This means that there are uncountably many possibilities for a policy. A solution to this problem is to discretize the time into equally sized intervals, and constrain policies to be constant during an interval. That is to say, such policies are only allowed to make a selection at the beginning of an interval, and may not select another immediate transition during an interval. They must stick to their decision made at the beginning of the interval. Such finite piecewise continuous policies approximate optimal timed policies arbitarily closely by tuning the granularity of the discretization. A possible way to achieve is to discretize an IMC (or, equivalently, a MA), and solve such MDP-like processes using value iteration [22]. Other recent techniques use uniformisation [5] or more advanced techniques [18]. Experiments indicate that these procedures are time consuming. Further research is needed to advance computing time-bounded reachability objectives.

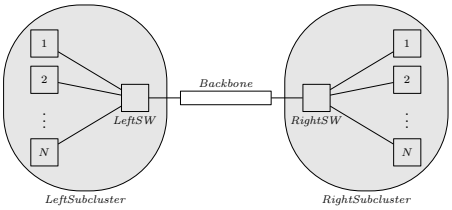

Fig. 4. A dependable workstation cluster with $2N$ workstations [13]

event	duration	event	duration
$LeftWSFail$	$500h$	$LeftWSRepair$	$0.5h$
$RightWSFail$	$500h$	$RightWSRepair$	$0.5h$
$LeftSWFail$	$4000h$	$LeftSWRepair$	$4h$
$RightSWFail$	$4000h$	$RightSWRepair$	$4h$
$BackboneFail$	$5000h$	$BackboneRepair$	$8h$

TABLE I
AVERAGE DURATIONS FOR COMPONENT FAILURES AND REPAIRS

IV. WORKSTATION CLUSTER EXAMPLE: REVISITED

To show the results of some initial experiments that were conducted, we consider a case study. The dependable workstation cluster [13] case study is depicted in Fig. 4. We consider two identical subclusters, each of which consists of $N \in \mathbb{N}$ workstations that are interconnected by a switch. Via their switches and a central backbone, the workstations in the two subclusters can communicate with each other. Each component can fail, and there is a single repair unit to turn failed components into operational mode. For the dependability analysis, we adopt the failure rates of the components which are given in [13], cf. Table I.

A. GSPN model

The workstation cluster is modeled as the GSPN depicted in Fig. 5. The first two rows represent the N workstations in the left and right subcluster, respectively. Each single workstation fails after $500h$ of operation, on average. Hence, we associate a failure rate of $\frac{1}{500}$ to each workstation. Thus, the timed transitions $LeftWSFail$ and $RightWSFail$ are marking dependent: If n tokens are in $LeftWSUp$, each of them fails with rate $\frac{1}{500}$. Therefore, the timed transition $LeftWSFail$ has rate $\frac{n}{500}$. The same reasoning applies for $RightWSFail$. Once a component has failed, a single repair unit is available that can repair one failed component at a time. Depending on the type of component, the repair operation takes different average times, cf. Table I.

Our model is basically the GSPN as proposed in [13] and as used in all of its studies so far, except that the inspection transitions (such as $RightSWInspect$ and $RightWSInspect$) are immediate rather than timed transitions with a very high rate. Accordingly, whenever the repair unit is available and different components have failed, the choice which component to repair next is nondeterministic. (In [13] this is probabilistic such that the resulting marking graph is a CTMC.) As a result, the GSPN of Fig. 5 is *confused*. Due to the presence of confusion, this net is not analyzable using the classical analysis techniques for GSPNs.

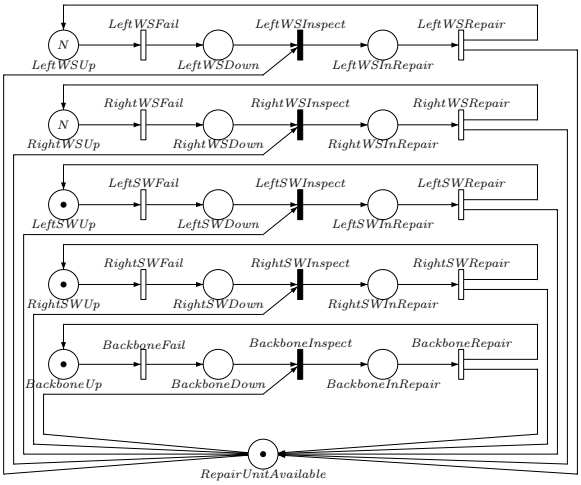

Fig. 5. GSPN model of the fault tolerant workstation cluster [13]

B. Expected time and long-run average analysis

Table II shows the computation times for the maximum expected reachability times where the set G of goal states depends on the number N of operational workstations. More precisely, G is the set of states in which none of the operational left (or right) workstations connected via an operational switch and backbone is available. For the sake of comparison, the next column indicates the computation times for unbounded reachability probabilities for the same goal set. The computation of these values is standard for MDPs and amounts to solving a linear programming problem. The last column of Table II lists the results for the long-run average analysis; the model consists of a single end component. The experiments were conducted on a single core of a 2.8 GHz Intel Core i7 processor with 4GB RAM running Linux.

C. Time-bounded reachability analysis

Let $k \in \{2, 3, \ldots, 2N\}$ be the number of workstations that are operational and mutually connected. Consider the following time-bounded reachability property: if the QoS constraint k is violated, the probability to face the same problem after d time units is at most p. We are interested in the maximal probability that the above event happens. It turns out that the dependability measures for the standard GSPN semantics (using uniform weights) and our stochastic game analysis differ considerably: In the worst case, the dependability is 18% worse than predicted by the classical GSPN model! This substantial difference is explained as follows. Assume s_{down} is the marking where both switches, the backbone and all N workstations in the right subcluster have failed, whereas in the left subcluster, all workstations are operational. To compute the maximal probability, we have to consider the worst possible repair strategy. Therefore, note that if $k \leq N$, repairing the left switch establishes QoS. Thus, the desired worst case probability is obtained if all workstations in the right subcluster are repaired –which does not establish QoS–

N	# states	# transitions	$\lvert G \rvert$	$eT^{\max}(s, \Diamond G)$ time (s)	$\Pr^{\max}(s, \Diamond G)$ time (s)	$\text{LRA}^{\max}(s, G)$ time (s)
1	111	320	74	0.0009	0.0061	0.0046
4	819	2996	347	0.0547	0.0305	0.1137
8	2771	10708	1019	0.6803	0.3911	1.3341
16	10131	40340	3419	10.1439	5.3423	20.0278
32	38675	156436	12443	292.7389	94.0289	455.4387
52	100275	408116	31643	3187.1171	1807.7994	OOM

TABLE II
EXPECTED TIME AND LONG RUN AVERAGE RESULTS

N	k	states	d	results	
				new	classical
4	3	820	20	0.3797	0.3038
4	5	820	20	0.4219	0.3717
4	8	820	20	0.4278	0.4250
8	3	2772	10	0.9319	0.7457
8	10	2772	10	0.9805	0.9178
8	16	2772	20	0.6147	0.6089

TABLE III
RESULTS OF TIME-BOUNDED REACHABILITY

before repairing the left switch. This gives rise to the relatively low worst case probabilities. In the classical GSPN model, however, each immediate transition has weight one. Therefore, the probability to repair the switch in the otherwise intact left subcluster is $\frac{1}{5}$. Obviously, this implicit strategy does not reflect the worst case repair strategy. (A marking-dependent weight assignment could alleviate this problem, but is non trivial to determine in advance by a modeller.)

V. EPILOGUE

We proposed to define the semantics of GSPNs using Markov automata, stochastic real-time games. This allows for the support of confused GSPNs and GSPNs with immediate subnets. The resulting semantics is truly simple and (as we believe) intuitive. The key is to interpret possible nondeterminism between immediate transitions *as is*. To support the quantitative analysis of such stochastic real-time games, we presented novel algorithms for expected time and long-run average objectives. Although initial experiments indicate their scalability, we think that further algorithmic improvements can improve their efficiency. Time-bounded reachability objectives are more time-consuming and deserve more investigations.

Acknowledgements.

I my co-workers: Dennis Guck (RWTH Aachen), Tingting Han (Oxford), Holger Hermanns (Saarland), Martin Neuhäusser (Siemens), and Lijun Zhang (DTU Lyngby).

REFERENCES

[1] Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. IEEE TSE 29(6), 524–541 (2003)
[2] Bertsekas, D.P., Tsitsiklis, J.N.: An analysis of stochastic shortest path problems. Mathematics of Operations Research 16(3), 580–595 (1991)
[3] Boudali, H., Crouzen, P., Stoelinga, M.: A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Trans. Dependable Sec. Comput. 7(2), 128–143 (2010)
[4] Bozzano, M., Cimatti, A., Katoen, J.P., Nguyen, V., Noll, T., Roveri, M.: Safety, dependability and performance analysis of extended AADL models. The Computer Journal 54(5), 754–775 (2011)
[5] Buchholz, P., Hahn, E.M., Hermanns, H., Zhang, L.: Model checking algorithms for CTMDPs. In: CAV. LNCS, vol. 6806, pp. 225–242. Springer (2011)
[6] Chen, T., Han, T., Katoen, J.P., Mereacre, A.: Model checking of continuous-time Markov chains against timed automata specifications. Logical Methods in Computer Science 7(1) (2011)
[7] Chiola, G., Donatelli, S., Franceschinis, G.: GSPNs versus SPNs: What is the actual role of immediate transitions? In: Petri Nets and Performance Models (PNPM). pp. 20–31. IEEE (1991)
[8] Chiola, G., Marsan, M.A., Balbo, G., Conte, G.: Generalized stochastic Petri nets: A definition at the net level and its implications. IEEE TSE 19(2), 89–107 (1993)
[9] Chiola, G., Franceschinis, G., Gaeta, R., Ribaudo, M.: GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic Petri nets. Performance Evaluation 24(1–2), 47–68 (1995)
[10] Coste, N., Garavel, H., Hermanns, H., Lang, F., Mateescu, R., Serwe, W.: Ten years of performance evaluation for concurrent systems using CADP. In: ISoLA. LNCS, vol. 6416, pp. 128–142. Springer (2010)
[11] Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: IEEE Symposium on Logic in Computer Science (LICS). pp. 342–351. IEEE (2010)
[12] Guck, D., Han, T., Katoen, J.P., Neuhäußer, M.R.: Quantitative timed analysis of interactive Markov chains. In: NASA Formal Methods Symposium (NFM). LNCS, vol. 7226, pp. 8–23. Springer-Verlag (2012)
[13] Haverkort, B.R., Hermanns, H., Katoen, J.P.: On the use of model checking techniques for dependability evaluation. In: Symposium on Reliable Distributed Systems (SRDS). pp. 228–237. IEEE (2000)
[14] Hermanns, H.: Interactive Markov Chains and the Quest for Quantified Quality, LNCS, vol. 2428. Springer (2002)
[15] Macià, H., Valero, V., Cuartero, F., Ruiz, M.C.: sPBC: A Markovian extension of Petri box calculus with immediate multiactions. Fundam. Inform. 87(3-4), 367–406 (2008)
[16] Marsan, M.A., Balbo, G., Chiola, G., Conte, G.: Generalized stochastic Petri nets revisited: Random switches and priorities. In: Petri Nets and Performance Models (PNPM). pp. 44–53. IEEE (1987)
[17] Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. John Wiley & Sons (1995)
[18] Rabe, M.N., Schewe, S.: Finite optimal control for time-bounded reachability in CTMDPs and continuous-time Markov games. Acta Inf. 48(5-6), 291–315 (2011)
[19] Sanders, W.H., Meyer, J.F.: Stochastic activity networks: Formal definitions and concepts. In: Formal Methods and Performance Analysis (FMPA). LNCS, vol. 2090, pp. 315–343. Springer (2000)
[20] Segala, R., Lynch, N.A.: Probabilistic simulations for probabilistic processes. Nord. J. Comput. 2(2), 250–273 (1995)
[21] Teruel, E., Franceschinis, G.: Well-defined generalized stochastic Petri nets: A net-level method to specify priorities. IEEE TSE 29(11), 962–973 (2003)
[22] Zhang, L., Neuhäußer, M.R.: Model checking interactive Markov chains. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 6015, pp. 53–68. Springer (2010)