

Performance Analysis of χ Models using Discrete-Time Probabilistic Reward Graphs

N. Trčka, S. Georgievska, J. Markovski¹, S. Andova², and E.P. de Vink

Department of Mathematics and Computer Science
 Technische Universiteit Eindhoven
 P.O. Box 513, NL-5600 MB Eindhoven, The Netherlands

Abstract—We propose the model of discrete-time probabilistic reward graphs (DTPRGs) for performance analysis of systems exhibiting discrete deterministic time delays and probabilistic behavior, via their interpretation as discrete-time Markov reward chains. We build on the χ environment, a full-fledged platform for qualitative and quantitative analysis of timed systems based on the modeling language χ . The extension proposed in this paper is based on timed branching bisimulation reduction followed by a tailored inclusion of probabilities and rewards. The approach is applied in an industrial case study of a turntable drill. The resulting performance measures are shown to be comparable to those obtained by existent methods of the χ environment, viz. simulation and continuous-time Markovian analysis.

I. INTRODUCTION

The χ language [1] is a modeling language for control and analysis of industrial systems (machines, manufacturing lines, warehouses, factories, etc.). It has been successfully applied to a large number of industrial cases, such as a car assembly line, a multi-product multi-process wafer fab [2], a fruit juice blending and packaging plant [3], and process industry factories [4]. Initially, χ came equipped with features for the modeling of discrete-event systems only, and was not supported by a formal semantics. Recently, it has been redesigned and converted to a formal specification language [5]. At present, χ can be characterized as a process algebra with data. In addition, it was extended to handle both discrete-event and continuous aspects, allowing for the modeling of hybrid systems [1].

Originally, simulation was the only means to analyze χ models. For the verification of functional requirements, however, simulation renders insufficient. Therefore, a new approach has been taken, connecting χ to state-of-the-art verification tools and techniques. Currently, a χ model can be compiled to the input language of a number of model checkers, including SPIN [6], [7], μ CRL [8], [9] and UPPAAL [10], [11] (see Fig. 1). The translated model can subsequently be checked against the functional properties formulated in the target setting. Successful verification is usually succeeded by performance analysis and design optimization. At present, performance analysis of a χ model can be carried out either by simulation, or by analysis of the underlying continuous-time Markov (reward) chain [12], CTMC for short (see Fig. 2).

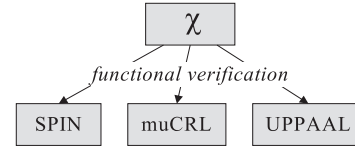


Fig. 1. Verification environment for χ

Simulation is a powerful method for performance analysis, but its disadvantages in comparison to analytical methods are well-known [13]. The approach based on CTMC turns χ into a powerful stochastic process algebra in the vein of [14], [15]. It is analytical, and builds on a vast and well-established theory. However, the generation of a CTMC from a χ model requires that all delays in the system are exponentially distributed. This is a serious drawback since in industrial systems, particularly in controllers, delays are often closer to being deterministic. It is possible to approximate deterministic delays by sequences of exponential delays, i.e. to model them by so-called phase-type distributions [16], but this approach suffers from the state explosion problem. Many states are needed to correctly approximate these delays, and the generated CTMC becomes large due to the full interleaving of stochastic transitions in parallel contexts.

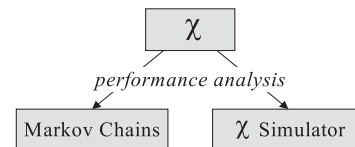


Fig. 2. Current performance analysis environment for χ

In this paper, we propose a model in which time delays are discrete and deterministic, while random behavior is expressed in terms of immediate probabilistic choices. This model is referred to as *discrete-time probabilistic reward graphs*, DTPRGs for short. DTPRGs can be viewed as a specialization of Semi-Markov Reward Chains. We define a method for obtaining performance measures of a DTPRG by transforming it to a discrete-time Markov reward chain [17], abbreviated as DTMRC. We augment the χ environment so that for a given χ specification, the corresponding DTPRG can be obtained automatically. Usually, in contrast to the CTMC

¹Supported by Bsik project BRICKS AFM 3.2.

²Corresponding author, e-mail s.andova@tue.nl.

approach, the DTPRG generated from a χ -model is considerably smaller (more than threefold for our case study). In a DTPRG, time itself does not decide a choice and, as such, interleaving of timed transitions does not occur as in typical timed process algebras [18]. As an illustration, a case study is discussed on the performance of a turntable drilling system. Although compact, this system is realistic and incorporates many complex modeling issues. The case has been studied previously to illustrate the verification techniques of functional requirements [5], [19]. We put the new performance results exploiting DTPRGs in perspective, by comparing them to results from simulation and the approach exploiting CTMCs.

II. DTPRGs

In this section we introduce the notion of a DTPRG, and give, regarding performance, two equivalent Markovian interpretations: one straightforward and general; the other more specific, but computationally more efficient.

Definition 1: A DTPRG is a tuple $G = (S, \rightsquigarrow, \mapsto, \rho)$, where (1) $S = S' \cup S''$, for two finite disjoint sets S' and S'' of *probabilistic and timed states*, respectively, (2) $\rightsquigarrow \subseteq S' \times (0, 1] \times S$ is a *probabilistic transition relation*, (3) $\mapsto \subseteq S'' \times \mathbb{N}^+ \times S$ is a *timed transition relation* such that $(s, n, s'), (s, m, s'') \in \mapsto$ implies $n = m$ and $s' = s''$, and (4) $\rho: S \rightarrow \mathbb{R}$ is a *reward rate assigning function*.

The interpretation of a DTPRG is as follows. In probabilistic states the process spends no time, and it jumps to a next state chosen according to the probabilistic transition relation. In a timed state the process spends as many time units as specified by the timed transition relation, and jumps to the unique subsequent state. The uniqueness requirement is to support the time-determinism property of [18]. A reward is gained per time unit, as determined by the reward rate assigning function. Although we allow reward rates to be assigned also to probabilistic states, the process actually gains no reward as it spends no time in them. We use infix notation and write, e.g., $s \xrightarrow{p} s'$ rather than $(s, p, s') \in \rightsquigarrow$.

We visualize a DTPRG as in Fig. 3a. For this DTPRG, states 1, 2, and 3 are timed, whereas states 4¹ and 5 are probabilistic. The reward rates are put in italics at the top right corner of each state; the reward rate of the state i is r_i , for $1 \leq i \leq 5$.

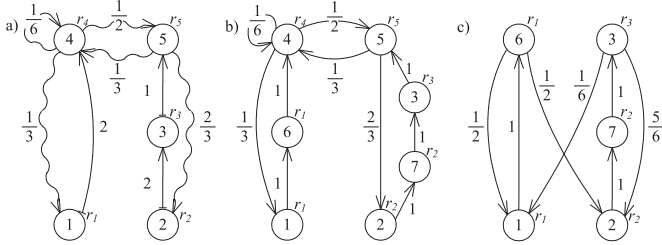


Fig. 3. a) A DTPRG, b) its unfolding, and c) aggregated unfolding

¹We allow probabilistic self-loop for specification convenience

A. From DTPRG to DTMRC

Most performance measures that we aim to obtain can be standardly defined. For example, the percentage of time the system spends in some state, the accumulated reward up to some time, etc. To obtain these measures we exploit a translation from DTPRGs into DTMRCs, as the latter are well-established models for performance analysis.²

A DTMRC, discrete-time Markov reward chain, is a triple $M = (S, \rightarrow, \rho)$, where S is a finite set of states, \rightarrow is a probabilistic transition relation over S , and ρ is a reward assigning function.³ Operationally, a DTMRC is considered to wait one time unit in a state, gain the reward for this state determined by the function ρ , and immediately steps to another state with a probability specified by the relation \rightarrow . When required by the context, we will have occasion to represent a DTMRC as a pair (P, ρ) , where P is the probability transition matrix and ρ is the state reward vector. For details of DTMRCs, we refer to the standard literature (e.g. [17]).

The main idea behind the translation from a DTPRG into a DTMRC is to represent a timed transition of duration n in the DTPRG as a sequence of n states in the DTMRC, connected by transitions labeled with probability 1, all having the same reward. Probabilistic transitions remain unchanged. We refer to this transformation as the *unfolding* of a DTPRG.

Definition 2: Let $G = (S_G, \rightsquigarrow, \mapsto, \rho_G)$ be a DTPRG with $S_G = \{s_1, \dots, s_n\}$. Associate with every state $s_i \in S_G$ a number $m_i \in \mathbb{N}^+$ as follows: if s_i is a probabilistic state, then $m_i = 1$; if s_i is a timed state, then $m_i = m$ for the unique m such that $s_i \xrightarrow{m} s_k$, for some $s_k \in S_G$. The *unfolding* of G is the DTMRC $M = (S_M, \rightarrow, \rho_M)$ with $S_M = \{s_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m_i\}$, $\rho_M(s_{ij}) = \rho_G(s_i)$, $s_{ij} \xrightarrow{1} s_{i,j+1}$ for $1 \leq j \leq m_i - 1$, and $s_{im_i} \xrightarrow{1} s_{k1}$ if $s_i \xrightarrow{m} s_k$ or $s_{i1} \xrightarrow{p} s_{k1}$ if $s_i \rightsquigarrow s_k$.

In order to stress the correspondence, the states in the DTMRC that relate to timed states in the original DTPRG will be referred to as *timed states*. Similarly, for probabilistic states and probabilistic and timed transitions.

The unfolding of the DTPRG from Fig. 3a is given by the DTMRC depicted in Fig. 3b. The unfolded timed delays originating from states 1 and 2 introduce the new states 6 and 7, respectively.

Note that the DTMRC obtained by unfolding, in general, does not truthfully represent the semantics of the original DTPRG, in the sense that probabilistic states are immediate in the DTPRG, whereas they take one unit of time in the DTMRC. For example, in the DTPRG from Fig. 3a, state 5 can be reached from state 1 with probability $\frac{1}{2}$ after a delay of 2 time units (via $1 \xrightarrow{2} 4 \xrightarrow{1/2} 5$), whereas in the unfolded version this cannot be done in less than 3 time units (that are required for a sojourn in the states 1, 6 and 4). The solution to this problem is to eliminate the immediate probabilistic states appropriately. This elimination is achieved by the aggregation

²For reasons of practical applicability, we choose this approach instead of a translation to Semi-Markov Reward chains [20].

³Note that we abstract from the initial probability vector.

method initially developed in the setting of stochastically discontinuous Markov processes [21] and subsequently adapted in [22], [23] for continuous-time Markov chains with explicit probabilistic transitions. Intuitively, this method computes the probabilities of reaching one timed state from another and adjusts the delays. More specifically, the process of aggregation is as follows: In a DTMRC $M = (P, \rho)$ the transition probability matrix P is represented as $P = P_t + P_p$, where P_t holds the unfolded timed transitions and P_p holds the immediate probabilistic transitions. Next, the Cesaro sum of P_p is computed and its canonical product decomposition (L, R) is found. The *aggregated* chain is defined by $\hat{M} = (LP_tR, L\rho)$.

The DTMRC in Fig. 3c is the aggregated chain of the one in Fig. 3b. The aggregation ‘removes’ the probabilistic states 4 and 5 and ‘re-establishes’ transitions between 6 and 1 and 2, according to the probabilities of the outgoing transitions of 4 and 5. Thus, in the aggregated chain there are two outgoing transitions from 6 to 1 and 2 (instead of a single one in the unfolded chain). This conforms to the Markovian semantics, that after a delay of one time unit there is an immediate probabilistic choice. It can be checked that this DTMRC represents a system with the same behaviour as the DTPRG in Fig. 3a.

B. Performance metrics

With the transformation of a DTPRG into a DTMRC in place, we can use the standard theory and tools to compute all common performance measures. For the present paper however, we focus on the long-run behavior of systems and on one particular measure called the *long-run expected reward rate* (long-run reward for short). The latter measure supports a number of interesting performance properties. If the resulting DTMRC is ergodic,⁴ the long-run reward is standardly computed as $R = \pi \rho$, where π is the long-run probability vector (in Cesaro sense), and ρ is the state reward vector. The full process of obtaining the performance measures of a DTPRG is visualized by the left branch in Fig. 4.

The performance measure of the DTPRG depicted in Fig. 3a is thus obtained by computing the long-run probability vector π of the DTMRC from Fig. 3c. This vector is $\pi = (\frac{1}{11} \frac{3}{11} \frac{3}{11} \frac{1}{11} \frac{3}{11})$, where states 6 and 7 are renamed in the matrix notation to states 4 and 5. The reward vector ρ equals $(r_1 \ r_2 \ r_3 \ r_1 \ r_2)$, so $R = \frac{2}{11}r_1 + \frac{6}{11}r_2 + \frac{3}{11}r_3$.

C. Optimization by geometrization

Note that the unfolded DTMRC has, in general, substantially more states than the original DTPRG, as every delay of duration n introduces $n-1$ new states. This means that the unfold & aggregate method, although straightforward to serve as a definition, leads to computations on large state spaces. In the rest of this section, we optimize our method, using

⁴Note that in case the resulting process is not ergodic, we can always partition the original DTPRG into subgraphs that produce ergodic processes and analyze them separately. So, we do not consider this as a limitation of our analysis.

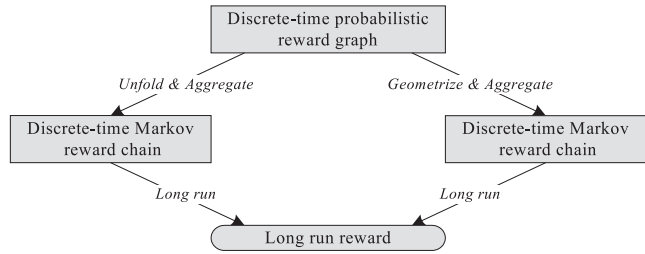


Fig. 4. Performance measuring for DTPRGs

‘geometrization’ of time delays to obtain a DTMRC of, at most, the size of the original graph. The main idea is to replace discrete delays by geometrically distributed delays instead of unfolding them.

Definition 3: A DTPRG $G = (S, \rightsquigarrow, \mapsto, \rho)$, is *geometrized* to the DTMRC $M = (S, \rightarrow, \rho)$, if (1) for each timed transition $s \xrightarrow{n} s'$ in G we have $s \xrightarrow{1/n} s'$ and $s \xrightarrow{(n-1)/n} s$ in M ; and (2) for each probabilistic transition $s \rightsquigarrow s'$ in G we have $s \xrightarrow{p} s'$ in M .

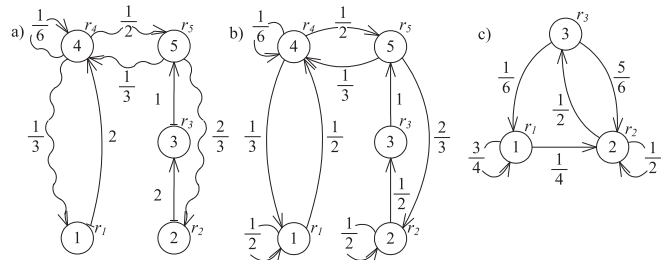


Fig. 5. a) A DTPRG, b) its geometrization, and c) aggregated geometrization

Consider again the DTPRG from Fig. 3a, repeated in Fig. 5a. In Fig. 5b its geometrized DTMRC is shown. For the same reason as before, this DTMRC still needs to be aggregated; the result is depicted in Fig. 5c.

The geometrize & aggregate method is depicted by the right branch in Fig. 4. The following theorem justifies the figure by showing that the two methods indeed commute, i.e. both give DTMRCs with the same long-run rewards.⁵

Theorem 1: Let M_1 be the unfolded & aggregated DTMRC, and M_2 the geometrized & aggregated DTMRC of the same DTPRG. Let R_1 and R_2 be the long-run rewards of M_1 and M_2 , respectively. Then $R_1 = R_2$.

The proof of the theorem (with the required preliminaries) is given in [24]. Here, we illustrate the result by an example. The long-run probability vector π' of the DTMRC in Fig. 5c is $\pi' = (\frac{2}{11} \frac{6}{11} \frac{3}{11})$. Its reward vector is $\rho' = (r_1 \ r_2 \ r_3)$, and so its long-run reward $R' = \frac{2}{11}r_1 + \frac{6}{11}r_2 + \frac{3}{11}r_3$ coincides with the R of the DTMRC from Fig. 3c.

⁵Note that the geometrization method can not, in general, be applied for non-long-run analysis.

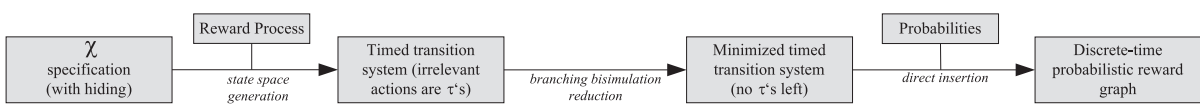


Fig. 6. χ to DTPRG

III. CASE STUDY: TURNTABLE DRILLING SYSTEM

In this section, we exploit DTPRGs in an industrial case study by measuring performance of a small drilling system. We first explain the system and how it is modeled in χ . We then show how to extend the χ environment to support and in particular generate, DTPRGs. Finally, following the methods discussed above, we calculate some relevant measures such as throughput and utilization of the system.

A. Description of the system

The turntable drilling system is a concrete example of a small but realistic manufacturing system [5], [19]. Its purpose is to make holes in products. The system consists of a round turntable and devices for adding, drilling, testing, and removing a product. The turntable has four slots and transports the products counterclockwise (see Fig. 7). The drilling device consists of a drill and a clamp. The drill makes a hole in the product, whereas the clamp is used to lock the product while drilling. The testing device measures the depth of the hole in the drilled product. If it reaches its down position, the test result of the product is positive. In that case the product is removed in the next rotation. Otherwise, it stays in the system to be drilled again. The turntable can treat up to four products at the same time, doing the operations in parallel.

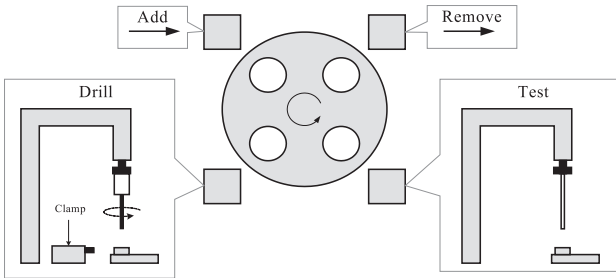


Fig. 7. The turntable drilling system

The various operations are modeled to require each a fixed amount of time. The system takes 3 time units to add a product, 2 time units to remove a product. The clamp needs 2 time units to lock or unlock a product. The drilling operation takes 3 time units, returning the drill to its up position takes 2 time units. Testing and returning the tester to its initial (up) position require 2 time units each.

For performance analysis, we make the assumption that the adding and the drilling are successful with a certain probability: when the system is about to add a new product, one is available with a certain probability; also, a product is drilled correctly with a certain probability.

B. χ model

For an introduction to χ , we refer to [1], [5]. Here, we only illustrate the features of the language by presenting part of the χ specification of the control system for the testing device:

```
Tester_Control( cStartTest, cTesterUpDone,
               cTesterDownDone, cTested,
               cTesterUpDown : chan ) =
  [| x, TstRes: bool |
   *( cStartTest?x;
     cTesterUpDown!true;
     ( cTesterDownDone?TstRes |
       delay 2.5; TstRes:=false );
     cTesterUpDown!true;
     cTesterUpDone?x;
     cTested!TstRes ) |]
```

The process `Tester_Control` receives a command to perform testing from the main control via the channel `cStartTest`. It then instructs the tester to go down via the channel `cTesterUpDown`. If the tester has reached its down position within 2.5 time units (recall that it needs 2 time units if the hole is properly drilled), the test of the product is considered successful (the input action `cTesterDownDone?TstRes` sets the boolean `TstRes` to true). If the sensor does not react within the 2.5 time units available, the controller marks the test result as bad. This timeout is modeled by means of the non-deterministic choice ‘|’ and uses the time deterministic semantics of χ according to which alternatives must always delay together. At the end, the controller instructs the tester to go up, along the channel `cTesterUpDown`, waits for the acknowledgement over the channel `cTesterUpDone`, and sends the test result to the main control via the channel `cTested`. The cycle then repeats itself which is modeled by the iterative construct ‘*’.

C. From χ to DTPRG

The standard semantics of (discrete-event) χ is in terms of timed transition systems [1], [25]. The main idea underlying the construction of a DTPRG from a timed transition system, as proposed here, is to hide all actions, i.e., to rename them to the special internal action τ , and then use the concept of timed branching bisimulation [18], [23] to reduce the system while abstracting from its internal transitions. If there is no real non-determinism in the model, a timed transition system without any action labeled transition is obtained, i.e., a DTPRG without probabilistic transitions. If there is one or more non-deterministic transition left, the system is underspecified, and its performance cannot be measured in the standard way.

Since χ has no features to model probabilistic choice,⁶ the random behavior of the adding and drilling devices is modeled

⁶Strictly speaking, it has, but only for simulation purposes.

in χ by a non-deterministic choice. When the corresponding DTPRG is generated from the χ model these non-deterministic choices must be appropriately replaced by probabilistic ones. For this we slightly adjust the method described in the previous paragraph. Instead of hiding all actions, the special actions used to indicate probabilistic branching remain visible. After the minimization, the probabilities that were intentionally left out are put as labels on the non-deterministic transitions. Again, if there is still non-determinism remaining in the model, we cannot proceed with performance analysis. Note that although the method is not always sound (in case of multiple probabilistic transitions from the same state) as it requests manipulation on the resulting graph, it serves its purpose for this and similar examples. Of course, another approach is to extend χ with an explicit probabilistic choice operator (e.g. the one in [26]). However, this requires drastic changes of the language and tools, and as such goes beyond the scope of this paper.

For the turntable system the reliability of the drill is captured in the tester process by a non-deterministic choice between sending the signal along the channel `cTesterDownDone` or doing the dummy silent action `skip`. Similarly, the availability of a product is taken care of in the process modeling the adding device.

The χ language does not directly support reward specification. We take a similar approach as for the absence of a probabilistic choice, and add rewards by manipulating the χ specification (again side-stepping changes in χ). We add, for each reward criterion, an ever repeating parallel component to the specification. The result is that in the timed transition system yielded, every state has a self-loop labeled by a special action denoting the reward rate of the state. These actions will not be hidden by branching bisimulation reduction and, therefore, persist in the resulting DTPRG. As in the case for the probabilistic choice, a systematic technique rendering the above can in principle be incorporated into the χ environment.

The complete pipeline of generating DTPRGs from χ specifications is illustrated in Fig. 6. Currently, we employ scripts tweaked into the χ environment that insert probabilities and rewards, in order to automatically produce the desired DTPRG from a given χ specification.

D. Performance analysis of the drilling system

We perform quantitative analysis of the turntable drilling system by applying the method proposed, and we compare the results with those obtained from simulation and CTMC analyses. We consider the following performance measures: (1) throughput, i.e. the number of products that leave the system per time unit; (2) utilization of the drilling machine, i.e. the percentage of time that the drill is actually drilling; and (3) the average number of products in the system. All measures are considered in the long-run.

In order to obtain the above measures, we assign rewards as follows. For throughput, we put the reward rate of $\frac{1}{2}$ only to the states in which the removal operation is performed. This is because in 2 time units, 1 product is removed from the system.

To obtain utilization, we give a reward rate of 1 to the states where drilling is performed. Finally, for the average number of products, every state is given a reward rate equal to the number of products present in the system when residing in that state.

The result of performance analysis is presented in Fig. 8, where each measure is represented as a function of the reliability of the drill and the availability of products. In Fig. 9 we give a comparison to the results obtained by simulation and CTMC analysis, when the probability for availability of a product is set to 0.5. We note that the model required for the DTPRG has 19023 states before reduction, whereas the DTPRG itself has 164; the model required for the CTMC analysis has 65529 states that reduces to a CTMC with 360 states by the weak Markovian bisimulation reduction of [14].

Note that the CTMC analysis gives the worst performance measures. This is anticipated, because the expected value of the maximum of two exponential delays is greater than the expected values of both delays, which increases the average cycle length of the turntable system. In the context of this paper we shortened the simulation experiments. For that reason, the simulation results do not align perfectly with the ones of the DTPRG analysis.⁷

IV. CONCLUSION

We have introduced a mathematical model, called discrete-time probabilistic reward graphs, abbreviated DTPRGs, for the performance measuring of systems featuring deterministic delay and probabilistic choice. We have extended the χ -environment to a prototype that supports the new model, enabling an effective qualitative and quantitative analysis of timed systems within the same framework. We have illustrated our method for a turntable drilling device, a relatively small but realistic industrial system. The results are shown to be comparable to those obtained by other methods in χ .

As future work we schedule the extension of the χ language to fully support the developed theory, relieving the script-based short-cuts taken presently that intervene at a proper place in the tool environment. We foresee that this can be achieved by introducing a probabilistic choice operator, and by facilitating the assignment of rewards in the toolset.

Related Work. The modeling of deterministic-time systems with probabilities as Markov chains has been studied previously, for different settings, in Petri net theory (e.g. [27]), process algebras (e.g. [26]) and automata (e.g. [28]). Our method differs in its incorporation of rewards, and that it is based on timed branching bisimulation reduction combined with the aggregation method for elimination of immediate transitions in Markov chains.

Acknowledgement. We are grateful to Bas Luttik for extensive comments on the draft of the paper.

⁷For simulation, we used the batch means method (cf. [13]), where each experiment lasted for 5000 time units.

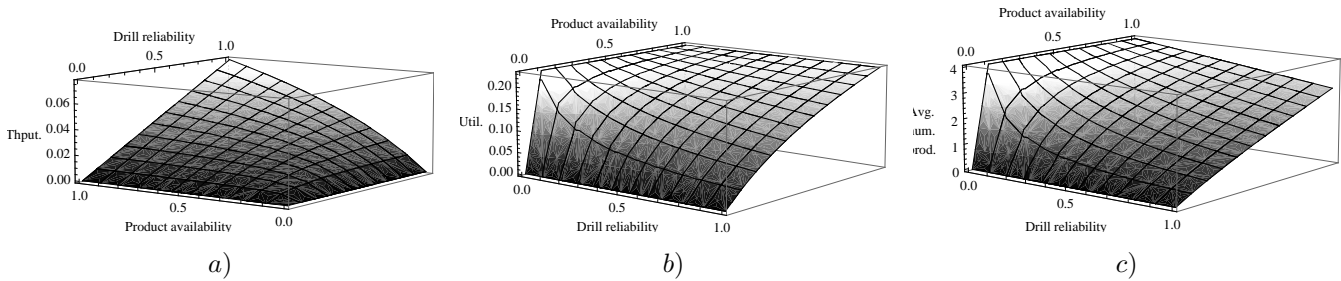


Fig. 8. a) Throughput, b) utilization, and c) average number of products of the turntable drilling system

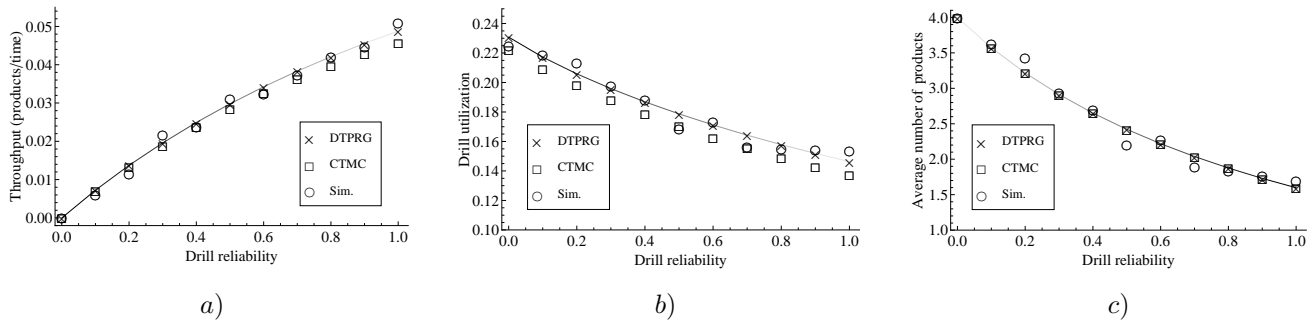


Fig. 9. Comparing the DTPRG method to CTMC analysis and simulation with product availability of 0.5

REFERENCES

- [1] D. A. van Beek, K. L. Man, M. A. Reniers, J. E. Rooda, and R. R. H. Schiffelers, "Syntax and consistent equation semantics of hybrid Chi," *Journal of Logic and Algebraic Programming*, vol. 68, pp. 129–210, 2006.
- [2] E. J. J. van Campen, "Design of a multi-process multi-product wafer fab," Ph.D. dissertation, Eindhoven University of Technology, 2000.
- [3] J. J. H. Fey, "Design of a fruit juice blending and packaging plant," Ph.D. dissertation, Eindhoven University of Technology, 2000.
- [4] D. A. van Beek, A. van der Ham, and J. E. Rooda, "Modelling and control of process industry batch production systems," in *IFAC'02*, Barcelona, 2002.
- [5] V. Bos and J. J. T. Kleijn, "Formal specification and analysis of industrial systems," Ph.D. dissertation, Eindhoven University of Technology, 2002.
- [6] G. J. Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, vol. 23, pp. 279–295, 1997.
- [7] N. Trčka, "Verifying χ models of industrial systems in Spin," in *ICFEM'06*. LNCS 4260, 2006, pp. 132–148.
- [8] S. Blom, W. Fokkink, J. F. Groote, I. van Langevelde, B. Lisser, and J. C. van de Pol, " μ CRL: A toolset for analysing algebraic specifications," in *CAV 2001*. LNCS 2102, 2001, pp. 250–254.
- [9] A. J. Wijs and W. Fokkink, "From χ_t to μ CRL: Combining performance and functional analysis," in *ICECCS'05, Shanghai*. IEEE, 2005, pp. 184–193.
- [10] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *STTT*, vol. 1, pp. 134–152, 1997.
- [11] E. M. Bortnik, D. A. van Beek, J. M. van de Mortel-Fronczak, and J. E. Rooda, "Verification of timed Chi models using UPPAAL," in *ICINCO*, Barcelona, 2005, pp. 486–492.
- [12] V. G. Kulkarni, *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1996.
- [13] J. Banks, J. S. Carson II, B. L. Nelson, and D. M. Nicol, *Discrete-event system simulation*. Prentice Hall, 2000.
- [14] H. Hermanns, *Interactive Markov Chains: The Quest for Quantified Quality*. LNCS 2428, 2002.
- [15] J. Hillston, *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
- [16] M. F. Neuts, *Matrix-geometric solutions in stochastic models, an algorithmic approach*. John Hopkins University Press, 1981.
- [17] J. G. Kemeny and J. L. Snell, *Finite Markov chains*. Springer, 1976.
- [18] J. C. M. Baeten, J. A. Bergstra, and M. A. Reniers, "Discrete time process algebra with silent step," in *Proof, language, and interaction: essays in honour of Robin Milner*. MIT Press, 2000, pp. 535–569.
- [19] E. M. Bortnik, N. Trčka, A. J. Wijs, S. P. Luttik, J. M. van de Mortel-Fronczak, J. C. M. Baeten, W. J. Fokkink, and J. E. Rooda, "Analyzing a χ model of a turntable system using Spin, CADP and UPPAAL," *Journal of Logic and Algebraic Programming*, vol. 65, pp. 51–104, 2005.
- [20] R. Howard, *Semi-Markov and Decision Processes*. Wiley, 1971.
- [21] M. Coderch, A. Willsky, S. Sastry, and D. Castanon, "Hierarchical aggregation of singularly perturbed finite state Markov processes," *Stochastics*, vol. 8, pp. 259–289, 1983.
- [22] J. Markovski and N. Trčka, "Aggregation methods for Markov reward chains with fast and silent transitions," in *Proceedings of MMB 2008*, Dortmund, Germany, 2008, To be published.
- [23] N. Trčka, "Silent steps in transition systems and Markov chains," Ph.D. dissertation, Eindhoven University of Technology, 2007.
- [24] N. Trčka, S. Georgievska, J. Markovski, S. Andova, and E. P. de Vink, "Performance analysis of chi models using discrete-time probabilistic reward graphs," TUE, Tech. Rep. CS 08/02, 2008, available from <http://www.win.tue.nl/~jmarkovs/CS08-02.pdf>.
- [25] J. C. M. Baeten and C. A. Middelburg, *Process Algebra with Timing*. Springer, 2002.
- [26] H. A. Hansson, *Time and Probability in Formal Design of Distributed Systems*. Elsevier, 1994.
- [27] W. M. Zuberek, "Timed Petri nets and preliminary performance evaluation," in *ISCA'80*. ACM Press, 1980, pp. 88–96.
- [28] R. Segala, "Modelling and verification of randomized distributed real-time systems," Ph.D. dissertation, MIT, 1995.