

# EFFICIENT EVENT-DRIVEN PROXEL SIMULATION OF A SUBCLASS OF HIDDEN NON-MARKOVIAN MODELS

**Robert Buchholz, Claudia Krull, Graham Horton**

University of Magdeburg, Faculty of Computer Science  
PO Box 4120, 39106 Magdeburg, Germany

*robert@isg.cs.uni-magdeburg.de, claudia@sim-md.de (Robert Buchholz, Claudia Krull)*

## **Abstract**

The paper introduces a new event-driven state space-based analysis algorithm for hidden non-Markovian models (HnMMs). HnMMs have been developed recently to enable the analysis of hidden discrete stochastic systems based on their observable output, e.g. to determine the unobserved causes of observed defects. There are currently two known approaches for analyzing HnMMs: Proxel-based analysis is generally applicable, but very time consuming and therefore infeasible for most realistic models; the modified Forward solver is very fast, but restricted to models of Markov regenerative type, which is a harsh restriction. The approach presented here bridges the gap between these two algorithms. It adapts the constant time steps of the Proxel algorithm to the time intervals between two output symbols and on the other hand encodes history into the modified Forward solver, thereby eliminating the need for the models to be of Markov regenerative type. However, the event driven Proxel algorithm requires every transition to produce observable output. Performance experiments show that the algorithm can generate a speed-up of up to factor 50 compared to the general Proxel solver for this restricted class of models. This paper extends the range of HnMMs that can be analyzed feasibly. It is another step toward practical feasibility of HnMM analysis, making them more useful for practitioners in the industry.

**Keywords:** Discrete stochastic system, hidden model, HnMM analysis, simulation algorithm

## **Presenting Author's Biography**

Robert Buchholz obtained a Bachelor's degree in Computer Information Systems from the University of Wisconsin - Stevens Point (USA), and a Bachelor's as well as a Master's ("Diplom") degree in Computer Science from the Otto-von-Guericke University Magdeburg (Germany). He is currently a Ph.D. student at Magdeburg University and conducts research on practical applications of new discrete simulation techniques.



## 1 Introduction

Discrete stochastic models (DSMs) are widely used in the industry, especially in the fields of manufacturing and logistics. They allow to explore possible system behaviors in order to better understand the system, analyze the behavior of a known system under various circumstances, and enable the user to find ways to optimize the system, e.g. to make it faster or use up less space.

So far, however, few practitioners use the power of hidden DSM analysis techniques. These allow for the analysis of discrete stochastic systems for which the system specifications are known, but the actual runtime behavior is not. These systems generate some kind of observable discrete output on some events, but the internal behavior is not observable. The analysis techniques then determine the most likely internal system behavior that has produced a given output sequence and thereby reconstruct the likely internal behavior from the observed output trace.

These hidden DSM analysis techniques can be used to reconstruct previous behavior from incomplete information (e.g. in forensics, or to find differences between system specifications and actual system behavior). They are also known to be able to determine unobserved causes of defects [1].

The most well-known efficient approach to model and analyze hidden DSMs are Hidden Markov Models (HMMs) [2], which however can only operate on Markovian (i.e. memoryless) systems, limiting the applicability. HMM have been extended in various ways to overcome some of their limitations (e.g. in [3]). These extensions slightly extend the modelling capabilities, but are still somewhat limited. Recently, however, Hidden non-Markovian Models (HnMMs) were introduced [4] to model and analyze arbitrary hidden discrete state systems. HnMMs, however, are currently computationally very expensive to analyze.

So, the analysis of HMMs is very fast, while HnMMs are very versatile, but to our knowledge there is no known modelling and analysis approach with extensive modelling capabilities *and* an efficient solution approach. Therefore, the goal of this work is to find a more efficient simulation algorithm for subclasses of HnMMs that are still more expressive than (modified) HMMs in order to be able to efficiently analyze less limited and therefore more realistic models. This will eventually enable us to analyze the hidden behavior of actual manufacturing systems, in order to gain insights not yet possible.

## 2 State of the Art

Hidden non-Markovian Models were introduced and classified in [5] based on the following three criteria:

- Whether all state transitions are reset after each symbol emission ( $T_{reset}$ ) or if some or all of them keep aging ( $T_{keep}$ ).

- Whether all ( $E_{all}$ ) or just some ( $E_{some}$ ) state transitions emit observable symbols.
- Whether a state can be reached from any other state through at most one ( $SC_{oneT}$ ) or an arbitrary number of state transitions ( $SC_{nT}$ )

The last dimension affects only certain implementation details, but has no influence on the effort necessary to analyze the model. It is therefore not examined any further in this work. Models may thus be appropriately classified using the other two dimensions, spanning a  $2 \times 2$  matrix of HnMM classes. The HnMM class of a model is determined by the modelling capabilities it requires and can potentially have a substantial impact on the computational effort necessary to analyze the model.

		Aging Behavior	
		$T_{Reset}$	$T_{Keep}$
Symbol Emissions	$E_{All}$	Modified Forward	
	$E_{Some}$		Modified Proxel

Fig. 1 Classification of HnMMs along with the corresponding analysis algorithms.

Figure 1 shows the matrix of HnMM classes and the currently-used analysis algorithm for each: An efficient analysis algorithm exists for models of the  $(T_{reset}, E_{all})$  class. It is a modified version of the Forward algorithm used to analyze HMMs and usually takes only a few seconds even when analyzing long output traces (containing 3000 and more symbols). For all other classes, an algorithm based on the Proxel[6, 7] method is used[8]. This algorithm, however, is computationally expensive, requiring computation times of several minutes to several hours even for small models. This is due to a potentially huge number of intermediate states that need to be considered and are discarded later, when the next symbol emission is observed.

### 2.1 Modified HnMM Forward Solver

The HnMM Forward solver is based on the Forward algorithm for HMMs [9]. It exploits the restrictions imposed by the  $(T_{reset}, E_{all})$  class of HnMMs:

- Due to  $T_{reset}$ , the probability to change state at the time of symbol emission depends only on the associated probability distributions and the length of the time interval between emissions. The history before reaching the current state has no influence and needs not be retained. This corresponds to models of Markov regenerative type that regenerate after every state transition.

- Due to  $E_{all}$ , the system changes its internal state only when a symbol is emitted. Consequently, there is no activity in between symbol emissions that would need to be analyzed. Since symbol emission times are the only relevant events, these intervals between two symbol emissions will be used as the analysis time step.

The algorithm starts by initializing the probability of each system state at the beginning of simulation time (e.g. by dividing the probability equally between all states, or by assigning a probability of one to the known start state). It then iteratively computes the probabilities for all systems states in each time step. In detail, the probability to have been in state  $A$  at time  $t_n$  and to have moved to state  $B$  at time  $t_{n+1}$  under emission of symbol  $X$  is computed as the product of

- The probability of being in state  $A$  at time  $t_n$
- The probability to not have left state  $A$  during the interval  $[t_n, t_{n+1}]$
- The relative (to all other possible state changes) probability to change the state from  $A$  to  $B$  exactly after  $t_{n+1} - t_n$  time has passed
- The probability to emit symbol  $X$  when changing state from  $A$  to  $B$

This product is then added to the probability to be in state  $B$  at time  $t_{n+1}$ .

The algorithm is very fast, but few models adhere to the restrictions of ( $T_{reset}, E_{all}$ ): that every change of the internal system state is observable, and that all processes are terminated and new ones restarted at every state change. Thus, most models cannot accurately be modeled using this approach.

## 2.2 Modified Proxel Solver

The modified Proxel solver was developed to analyze every imaginable kind of HnMM. It does not impose additional limitations beyond those of the definition of HnMMs. In particular,

- The model does not have to adhere to  $T_{reset}$ , so that the state change probability may depend not only on the length of the time interval between two successive symbol emissions, but also on the duration that each transition has been active since it last fired. Therefore, records of all possible combinations of these ages along with the corresponding probability must be kept.
- The model is not necessarily  $E_{all}$ , meaning that the system may change its state at any time (and even multiple times) in between two symbol emissions and these state changes are not observable. Hence, the algorithm must analyze all possible state changes between symbol emissions.

So, further possible system development depends on

- the current discrete system state
- the ages (duration of inactivity since last deactivation or firing) of all active and/or race-age transitions
- the probability of that state/age combination

and this information is stored for each possible system state as a so-called Proxel. The analyzing algorithm initializes the set of possible system states at the beginning of simulation time (usually by creating a single start Proxel with probability one). It then divides the simulation time into equally-sized (user-definable) time steps and evaluates all possible outcomes after each time step. It does so by computing successor Proxels to each Proxel of the current time step for all possible single state changes as well as for inactivity (no state change at all) during a time step. If two successors resulting from different Proxels have the same discrete system state and identical age vectors, their Proxels can be merged by simply summing up their probability. The original algorithm was first introduced in [6] and is described in depth in [7].

This approach is slightly modified to analyze HnMMs: For time steps that include a symbol emission, the same formula as for the modified Forward algorithm is used. This also means that possible events happening between a symbol emission and the end of the time step during which the emission happened are ignored. Additionally, all Proxels whose cause of creation is not in line with the observations from the output trace are discarded. This affects Proxels that were created through inactivity during emission time steps as well as those created by a symbol-emitting state transition during time steps where no symbol was emitted according to the output trace.

This algorithm is applicable to all HnMMs, but the often exponential growth in the number of possible system states (called “state-space explosion”) makes it impractical to analyze most HnMMs with this approach. Thus, it is not computationally feasible to use this approach to simulate the majority of models.

## 2.3 Summary

So, two algorithms exist to analyze Hidden non-Markovian models: An efficient modified Forward algorithm that is applicable only to a small subset of useful HnMMs; and the modified Proxel approach that can be used to analyze any HnMM, but is not necessarily the most efficient choice for a given model and may often be computationally infeasible.

The modified Proxel approach was developed for the least strict interpretation of HnMMs ( $E_{some}, T_{keep}$ ) and therefore works also for more restricted classes of HnMMs. However, it stands to reason that for these more restricted classes, more efficient algorithms may be developed.

### 3 New Event-Driven Proxel Approach

The main contribution of this work is the development of an efficient event-driven computation algorithm for the  $(E_{all}, T_{keep})$  case. Even though this class has some limitations over the most general  $(E_{some}, T_{keep})$  HnMM class, there are practically relevant models that fall into this class, for example the one analyzed in [1].

For this class of HnMMs:

- $E_{all}$  holds and thus the times of all state changes are known and the algorithm can analyze these events separately from the known time interval of inactivity in between (hence the name “event-driven”).
- the generic  $T_{keep}$  is assumed. Therefore, history has an impact on possible future system developments and must be retained. The forward HnMM solver is thus not applicable for this class of models.

The new approach occupies a middle ground between the fast Forward solver and the versatile Proxel solver. It can be seen either as an extension to the forward solver, or as a simplification of the Proxel solver. For the purpose of describing the algorithm, the latter approach will be used. For sake of completeness, the former will be mentioned briefly afterwards.

#### 3.1 Motivation

For the class of  $E_{all}$  models, all state transition times are known in advance. When analyzing these models with the modified Proxel solver, it can be observed that during time steps without symbol emissions, the only valid successor Proxels are those representing inactivity. No additional Proxels are created (or, equivalently, Proxels representing state changes are created and discarded at once).

So, the intervals of inactivity between symbol emissions are essentially reduced to a stepwise computation of the probability of inactivity for each Proxel (Figure 2 shows a schematic Proxel tree for this simulation approach). The time steps during which symbols

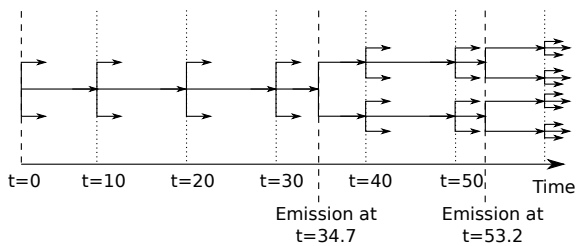


Fig. 2 Schematic development of the Proxel tree for a  $E_{all}$  HnMM using the modified Proxel solver. Arrows without successors represent Proxels that are being discarded.

are emitted contain both, inactivity up to the emission, the time of symbol emission itself, and system behavior

that could unfold after the emission till the end of the time step (which is ignored in this approach).

It would be more elegant to distinguish between periods of inactivity and times of symbol emission, and to compute these separately.

#### 3.2 Algorithm

So the idea of the new event-driven Proxel algorithm is to no longer use constant time steps, but instead to regard each interval of inactivity between two successive symbol emissions as a single time step. Consequently, these time steps will vary in length. They will also never contain a symbol emission, but will always end with one. Figure 3 shows a schematic Proxel tree

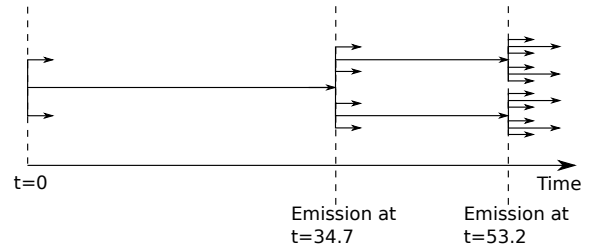


Fig. 3 Schematic development of the Proxel tree for the same  $E_{all}$  HnMM using the event-driven Proxel solver. Arrows without successors represent Proxels that are being discarded.

for this new approach: A successor Proxel will always be created as the result of a period of inactivity over the course of a whole time step, followed by a symbol emission caused by a state change.

The probability of inactivity during a time step is computed the same way as for the modified Proxel solver, by a suitable numerical integration of the differential equation

$$\frac{d\Pi}{dt} = -\Pi(t) \sum_i \mu_i(\tau_i(t)) \quad (1)$$

over the current time step. Here,  $\Pi(t)$  is the probability to still be in the current state at time  $t$ ,  $\mu_i$  is the hazard rate function associated with the probability distribution of the  $i$ th active state transition in the current state, and  $\tau_i(t)$  is the duration that the state transition  $i$  has been active at time  $t$ .

The relative probability to change state due to transition  $j$  firing at time  $t$ , under the condition that a transition is known to have fired at that time and has emitted a symbol, is also still computed by the same formula as in the case of the modified forward and Proxel solvers:

$$\Pi_{relative_j} = \frac{\mu_j(\tau_j(t))}{\sum_i \mu_i(\tau_i(t))} \quad (2)$$

The probability of being in each possible target state  $B$  at the end of a time interval when one has been in state  $A$  at the beginning of the interval and symbol  $X$  was emitted is then simply the product of

- the probability of inactivity during that time step
- the relative probability to change state from  $A$  to  $B$  at the end of the time step
- the constant given probability of emitting symbol  $X$  while changing state from  $A$  to  $B$

The pseudocode for the full algorithm is given in Algorithm 1.

---

**Algorithm 1:** Pseudocode for the event-driven Proxel solver.

---

**Data:** initialProxels, Trace  
**Output:** nextStepProxels = finalStateProxels  
currentProxels = initialProxels;  
**foreach** *TraceElement*  $el \in Trace$  **do**  
  nextStepProxels.clear();  
   $\Delta t = el.emissionTime - prevEmissionTime$ ;  
  prevEmissionTime = el.emissionTime;  
  **foreach** *Proxel*  $p \in currentProxels$  **do**  
    stayProb = computeInactivityProbability( $p$ ,  
     $\Delta t$ );  
    hrfSum = 0;  
    **foreach** *Transition*  $trans \in p.state.transitions$   
    **do**  
      hrfSum += trans.getHrfValue( $p.ages$ );  
    **foreach** *Transition*  $trans \in p.state.transitions$   
    **do**  
      transProb = trans.getHrfValue( $p.ages$ ) /  
      hrfSum;  
      pSucc = p.createSuccessor( $trans, \Delta t$ );  
      pSucc.probability = p.probability \*  
      stayProb \* transProb \*  
      trans.emitProbability( $el.symbol$ );  
      nextStepProxels.addOrMerge(pSucc);  
  currentProxels = nextStepProxels;  
**return** nextStepProxels

---

### 3.3 Consequences

These modifications from the old modified Proxel approach cause a number of consequences, most of which are beneficial:

Since the time interval between two symbol emissions can become quite big (relative to the support of the active transitions), simple single-step integration to compute the inactivity over each time interval no longer yields sufficiently accurate results. The computation should be based on an adaptive integration method. For our implementation, we use a modified version of the Dormand-Prince (ODE45) integration method [10].

No Proxels are created for the time between the emission of two symbols, making the algorithm more efficient. The analysis speed increases, since intermediate time steps need not be computed. This reduces the time needed to solve the inactivity ODEs, create the intermediate Proxels, and find potential Proxel merge candidates.

The approach does not introduce additional errors beyond those of the Proxel method. All simplifications

are justified by the properties of  $E_{all}$ . So the new event-driven approach is faster, but not less accurate than the modified Proxel approach. The approach can actually be more accurate than the Proxel solver, because the Proxels' age vector elements are no longer forced to be a multiple of the time step size, but can contain the correct ages. Furthermore, the event-driven solver does not suffer from the normal Proxel accuracy impact caused by effectively ignoring all system behavior between a symbol emission and end of the corresponding time step, since in the new algorithm, the symbol emission always marks the end of a time step.

Users of the event-driven Proxel approach are also not plagued by a side-effect of the HnMM Proxel analysis: Since the method can only simulate a single symbol emission per time step, time step sizes needed to be smaller than the smallest interval between two symbol emissions. This often meant a huge performance impact - since computation time increases exponentially with reduced step size - without a notable gain in accuracy. The event-driven Proxel simulation adapts time steps to the individual inter-symbol-emission intervals and thus does not have this limitation.

The limitations of  $E_{all}$  along with the algorithm's adaptation to them result in a far simpler analysis of the result errors. The Proxel method has three inherent sources of errors[11]: one related to only simulating a single state transition per time step, one caused by the integration inaccuracy, and one caused by expressing age vector elements representing state changes that happened anywhere during a whole time step as a single number, which furthermore must be a multiple of the simulation step size. The event-driven approach for  $E_{all}$  models accurately simulates the only state transition per time step and since these transitions can happen only at a certain point in time (as opposed to a time interval), the age values stored inside the Proxels of the event-driven method are also accurate. This leaves the only error to be the integration error. For this error, numerous techniques exist to reduce or measure it, making it a relatively simple task to assess the simulation accuracy for a given model.

Since the integration error can be controlled automatically by adaptive integration methods, no more user-tunable parameters are necessary, making the approach easier to use for practitioners.

### 3.4 Alternative Interpretation as Extension to the Forward HnMM Solver

The algorithm could also be interpreted as an extension to the Forward HnMM solver. For both classes of models (the one for which the Forward HnMM solver is applicable, and the one for which we developed the event-driven Proxel simulator)  $E_{all}$  holds and so state changes occur at known points in time. For both algorithms, time steps can be defined in such a way that the symbol emissions occur at the end of each time step. Then, the probabilities of inactivity during a time step and the relative probability of a given transition firing can be computed with the same formulas in both cases.

The reason that the modified Forward solver is not directly applicable to models of the  $(E_{all}, T_{keep})$  class is that through  $T_{keep}$ , history has an influence on inactivity and transition probabilities. For this class of models, the probabilities depend not only on the current system state, but also on how long the transitions have been active since they last fired or were deactivated.

In order for an algorithm of the Forward structure to work with  $T_{keep}$  models, this age information needs to be encoded into the system states. Thus, for the computation, each discrete state needs to be represented by a set of Markov Chain states, which are a combination of the discrete state with all possible reachable age vectors in that state.

The algorithm would then perform the same computations in each time step as the event-driven Proxel approach, and would arrive at the same result.

This alternative description of the development of the event-driven Proxel approach shows that the approach is not a radically different approach, but rather lies on a continuum between the already known algorithms. This notion suggests that it should be possible to develop similar algorithms for other subclasses of the HnMMs as well by finding a suitable middle ground between the fast Forward solver and the versatile Proxel solver.

## 4 Experiments

The purpose of this section is to detail performance comparison experiments conducted on  $(E_{all}, T_{keep})$  models with the event-driven Proxel solver and with the modified Proxel solver, since these are the only known algorithms, which can sensibly analyze models of that class.

Its goal is to determine, under which conditions a performance increase of the event-driven Proxel simulator over the modified Proxel solver can be expected. To that end, the following questions are to be answered:

- How does the accuracy of the two approaches differ when using the same simulation step size (integration step size in case of the event-driven algorithm) ?
- How does the accuracy of the two approaches differ when allowing for the same computation time?
- How do the computational costs of both approaches change when increasing the trace length?
- How does the accuracy and computation time of the event-driven Proxel solver change, when it is modified to not use the correct real-valued age values in its Proxels, but uses values rounded to a multiple of a given constant step size?

The model used in all experiments is shown as a stochastic Petri Net with additional symbol emission probabilities in Figure 4. It represents two machines that produce indistinguishable items with known inter-arrival times, and a downstream quality tester (which in

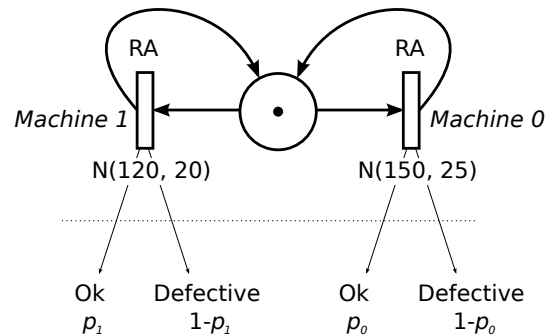


Fig. 4 The “Tester” model

the model causes the symbol emissions) that tests the produced items for defects. The tester logs the test result (“Ok” or “Defective”) and time stamp of each test, but cannot determine the source of the tested item. The task of an HnMM analyzer is therefore to determine an estimate of how many of the known-to-be-defective items were produced by each machine. Since the actual defective probabilities are not known (as they are to be computed), their are all set to 0.5 in the model.

### 4.1 Experiment Setup

All computation times recorded are the CPU times of the respective simulation processes, not the overall simulation duration (wall time). This approach reduces the influence that concurrently running programs may have on the computation time. All experiments were conducted on a Core2Duo 3GHz CPU. However, both simulation programs are single-threaded, using only one of the CPU cores. The CPU was manually set to always run at full speed in order to eliminate the influence of temporary CPU frequency reduction on the computation times.

The implementations of the modified Proxel solver and the event-driven Proxel solver algorithms share most of their source code. They use identical code for Proxel storage, numerical integration, and probability distributions. Different code is only used when the differences in the algorithms make it necessary. Consequently, the differences in computation time can indeed be attributed to the different theoretical approaches with great confidence.

### 4.2 Accuracy

To test the accuracy of the two approaches, a single sample output trace of the model was analyzed by both algorithms using *viable* simulation step sizes. Since the event-driven Proxel simulator does not operate on simulation steps directly, the selected step size was used as the upper limit for the integration step size of all probability computations. For the event-driven Proxel solver, all integration step sizes are viable. For the modified Proxel solver, a viable simulation step size is one where no two symbol emissions happen during a single time step. This limitation is necessary, because the modified Proxel solution algorithm for HnMMs is not defined for these cases.

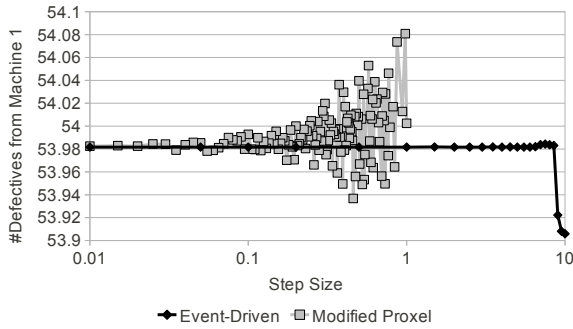


Fig. 5 Results for the experiment on algorithm accuracy for given step sizes. The horizontal axis has a logarithmic scale.

The results for the computed expected values for the number of defective items produced by machine 0 are shown in Figure 5. For decreasing step sizes, the results for both algorithms converge to the same value, indicating that both algorithms do indeed solve the problem correctly. However, the results for the event-driven algorithm converge much faster, agreeing with the final result in four digits already at a step size of 8, while the modified Proxel solver needs a step size of 0.2 for the same accuracy.

The difference is likely to be explained by assumptions the algorithms make on the system behavior: The modified Proxel solver ignores any events that happen between a symbol emission and the end of the corresponding time step. The event-driven algorithm starts a new time step after each symbol emission and thus does not make this assumption.

### 4.3 Simulation Efficiency

The simulation efficiency is the result accuracy that can be achieved by investing a given amount of computation time. Using the results from Section 4.2, the value that the results for both algorithms converge to for decreasing step sizes (about 53.982) can be assumed to be the correct simulation result.

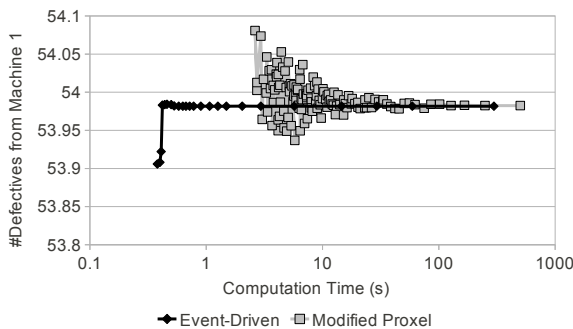


Fig. 6 Results for the experiment on algorithm efficiency. The horizontal axis has a logarithmic scale.

From the information gathered from the experiment in Section 4.2, the computation times and the computed

simulation results are taken and plotted in Figure 6. For the event-driven algorithm, the simulation results converge to the final result with only minor investment in computation time. For a computation time of only one second, the results differ by no more than 0.01 from the final result. Due to the limitation to *viable* step sizes, the modified Proxel solver is not able to compute a simulation result within that time frame. For the same accuracy, the modified Proxel solver needs about 50 seconds. Thus, this experiment demonstrated a gain in efficiency by a factor of 50 for the event-driven solution algorithm.

This gain is likely due to two factors. The first factor is the same as mentioned in Section 4.2: The Proxel method makes an additional assumption on the system behavior between a symbol emission and the end of a time step. This assumption generally does not hold and thus causes an additional error.

The second factor is related to the integration accuracy: The modified Proxel simulation needs to compute integration results for each time step to compute the transition probability and create the Proxels for the next time step. This forces the maximum permissible integration step size to be the simulation step size, even if such a small step size may not be necessary. This causes a performance penalty, particularly when other assumptions of the algorithm cause a bigger error, so that the higher integration accuracy cannot result in a more accurate final result. Thus, the modified Proxel solver has to perform many more computationally expensive integration steps than would actually be sensible. For the event-driven solver, each integration step size can be as big as the time between two symbol emissions and need only be reduced to actually increase accuracy, saving unnecessary computations.

### 4.4 Behavior Under Increasing Trace Length

The behavior under increasing trace length is one way of measuring the ability of an approach to cope with increasing complexity. This experiment was conducted in order to ensure that the favorable results obtained in the previous experiments are generalizable and do not apply to just a limited set of parameters.

For this experiment, the model was analyzed with both algorithms using only the first  $n$  elements of a trace. The simulation and integration step size was set to 0.25 in order to yield computation times that are high enough to not be impacted by random influences.

Figure 7 shows the results: both algorithms exhibit only a linear increase in computation time with increasing trace length, and both increase by almost the same factor. This behavior was to be expected. The event-driven simulation approach removes some redundancies in computing intermediate results of the modified Proxel approach, but after each symbol emission, both algorithms still contain almost the same number of Proxels. Since integration step size was fixed to make results comparable, the event-driven approach cannot benefit from using more generous integration step sizes. The slight overhead of the event-driven approach is

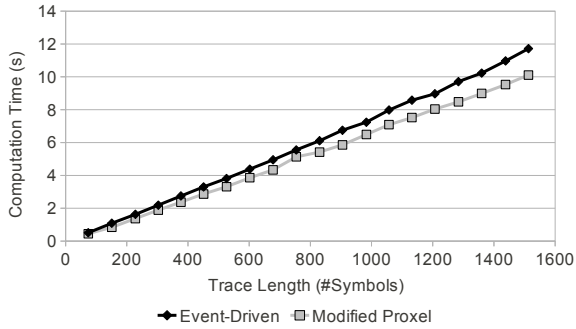


Fig. 7 Computation times for varying trace lengths for both algorithms. Simulation and integration step size was set to 0.25 in all cases.

likely due to the event-driven Proxels to contain arbitrary real-valued age vector elements (while the Proxels of the other algorithm contain only age values that are an integer multiple of the current step size) and thus not being able to merge as many Proxels as the old approach.

Overall, this experiment shows that the modified Proxel solver and the event-driven Proxel solver take almost the same amount of time to finish a computation for the same trace length and step size. Since the event-driven approach has been shown to achieve a higher accuracy within the same computation time, this means that it will keep that advantage under increasing trace lengths as well.

#### 4.5 Rounded Age Values

For the modified Proxel approach, the elements of a Proxel’s age vector are all multiples of the simulation step size. On one hand, this introduces an error, especially in cases where the event that determines an age value is known to have happened at an exact point in time. On the other hand, this restriction to a comparatively small set of age values means that successors from different Proxels are likely to have a common marking and age vector. Thus they can be merged, reducing the number of Proxels to be processed and thereby increasing the approaches efficiency.

For our event-driven Proxel approach based on known times of events, age values could and were computed exactly (within the limits of IEEE 754 floating point numbers). It is therefore of interest, how an artificial rounding of these values to multiples of a given step size may influence the method, both by decreasing its accuracy, but also by allowing more merging opportunities.

Therefore, we simulated our “Tester” model using first a normal event-driven Proxel approach as detailed in this paper. Then, we performed the same experiment using an event-driven Proxel simulator that was modified to artificially round the age values of all Proxels to the nearest multiple of the step size used for integration.

For each step size, the computation time of the two al-

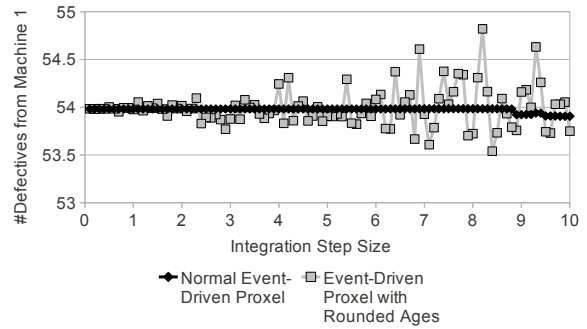


Fig. 8 Computed simulation result of the “Tester” model for different trace lengths.

gorithms did not differ by more than one percent and thus the data on this minor difference is not reproduced here. Apparently, the potential for merging Proxels was not increased through the rounding of the age values. Figure 8 shows the simulation result computed by both algorithms for different integration step sizes (and thus, different values to round the age values to). While the value computed by the usual event-driven Proxel converges quickly (i.e. for comparatively big step sizes) to a single result, the approach with introduced age rounding converges to the same value much slower, comparable to the standard Proxel approach (cf. Figure 5).

Thus, the rounding of age value negatively impacts the result accuracy strongly, without providing any benefits, at least for the model tested.

		Aging Behavior	
		$T_{Reset}$	$T_{Keep}$
Symbol Emissions	$E_{All}$	Modified Forward	Event Driven Proxel
	$E_{Some}$	?	Modified Proxel

Fig. 9 Extended classification of HnMMs along with the corresponding well-adapted analysis algorithms.

## 5 Conclusion

In this paper, we introduced an optimized solution algorithm for the analysis of the  $(E_{all}, T_{keep})$  subclass of Hidden non-Markovian Models. The experiments showed that the new event-driven Proxel approach can speed up the analysis of this well-defined and practically useful class of HnMMs remarkably.

This makes it less time-consuming to obtain analysis results for existing models. It also makes analysis of more complex and thus more realistic models possi-



ble, allowing insight into the internal behavior of these hidden systems for the very first time. Finally, the approach does not require any manual tuning – while the previously-used approach requires a manual tuning of the simulation time step – , making it less demanding to use by practitioners.

This increased feasibility of analyzing HnMMs is another step towards analyzing more complex and thereby more realistic models and thereby another step towards practical applicability.

### Future Work

As suggested in section 3.4, similarly efficient algorithms may exist for other subclasses of the HnMMs and are the subject of ongoing research. A particular focus is set on finding a more efficient analysis algorithm for the class of  $(E_{some}, T_{reset})$  (cf. Figure 9), because these models are currently analyzed with the modified Proxel HnMM solver, but exhibit additional restrictions that are not currently exploited.

## 6 References

- [1] Robert Buchholz, Claudia Krull, Thomas Strigl, and Graham Horton. Using hidden non-markovian models to reconstruct system behavior in partially-observable systems. In *3rd International ICST Conference on Simulation Tools and Techniques*, 2010.
- [2] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.
- [3] Felix Salfner. Modeling event-driven time series with generalized hidden semi-markov models. Technical report, Humboldt-Universität zu Berlin, 2006.
- [4] Claudia Krull and Graham Horton. The effect of rare events on the evaluation and decoding of hidden non-markovian models. In *Proceedings of the 7th International Workshop on Rare Event Simulation (RESIM 2008)*, Rennes, France, September 2008.
- [5] Claudia Krull and Graham Horton. Hidden non-markovian models: Formalization and solution approaches. In *Proceedings of 6th Vienna International Conference on Mathematical Modelling*, Vienna, Austria, February 2009.
- [6] Graham Horton. A new paradigm for the numerical simulation of stochastic petri nets with general firing times. In *European Simulation Symposium*, Dresden, Germany, October 2002. SCS European Publishing House.
- [7] Sanja Lazarova-Molnar. *The Proxel-Based Method: Formalisation, Analysis and Applications*. PhD thesis, Otto-von-Guericke Universität Magdeburg, 2005.
- [8] Claudia Krull and Graham Horton. Solving hidden non-markovian models: How to compute conditional state change probabilities. In *21st European Modeling and Simulation Symposium*, Santa Cruz de Tenerife, Spain, September 2009.
- [9] Gernot A. Fink. *Markov Models for Pattern Recognition*, chapter 7, page 121. Springer Berlin Heidelberg, 2008.
- [10] Claudia Krull, Robert Buchholz, and Graham Horton. Improving the efficiency of the proxel method by using individual time steps. In *The 16th International Conference on ANALYTICAL and STOCHASTIC MODELLING TECHNIQUES and APPLICATIONS*, pages 116–130. Springer-Verlag Berlin Heidelberg, 2009.
- [11] Robert Buchholz. Improving the efficiency of the proxel method by using variable time steps. Master’s thesis, Otto-von-Guericke Universität Magdeburg, 2008.