Computing Cumulative Rewards using Fast Adaptive Uniformisation

Frits Dannenberg[1], Ernst Moritz Hahn[1,2], and Marta Kwiatkowska[1]

[1] University of Oxford, Department of Computer Science,
[2] State Key Lab. of Comp. Sci., Institute of Software, Chinese Academy of Sciences, China

Abstract. The computation of transient probabilities for continuous-time Markov chains often employs uniformisation, also known as the Jensen's method. The fast adaptive uniformisation method introduced by Mateescu approximates the probability by neglecting insignificant states, and has proven to be effective for quantitative analysis of stochastic models arising in chemical and biological applications. However, this method has only been formulated for the analysis of properties at a given point of time t. In this paper, we extend fast adaptive uniformisation to handle expected reward properties which reason about the model behaviour until time t, for example, the expected number of chemical reactions that have occurred until t. To show the feasibility of the approach, we integrate the method into the probabilistic model checker PRISM and apply it to a range of biological models, demonstrating superior performance compared to existing techniques.

1 Introduction

Model checking of continuous-time Markov chains (CTMCs) [3] is an established method that has been successfully used for quantitative analysis of a variety of models, ranging from biochemical reaction networks [9,17] to performance analysis of computer systems [2]. The analysis typically involves computing the *transient probability* of the system residing in a state at a given time t, or, for a model annotated with time-dependent rewards, the *expected reward* that can be obtained. Transient probabilities for finite-state CTMCs can be computed through the *uniformisation* method, also known as the Jensen's method. Uniformisation involves discretising the CTMC with respect to a fixed rate, which enables reduction of the transient probability calculation to an infinite summation of Poisson distributed steps of the derived discrete-time Markov chain, and approximating the probability by truncating to a finite summation. The number of terms required can be precomputed for a specified precision using the Fox-Glynn method [8].

Many biochemical reaction networks, however, induce CTMC models whose state space is potentially infinite. To handle such cases, [17] introduced *continuous-time propagation models*, a generalisation of continuous-time Markov chains. The idea of this model is to propagate the (probability or expectation) mass values along the system execution. In order to analyse propagation models, the *fast adaptive uniformisation (FAU)* method was formulated [16]. Similarly to standard uniformisation, FAU applies discretisation, except that it does so dynamically, starting from some initial condition wrt to a sequence of rates (a birth process) rather than a single rate, and truncates the computation of the probability to a finite summation, although the number of summation terms cannot be precomputed. To deal with the unbounded state space, FAU explores only the relevant states, ignoring the probability of the insignificant states. Thus, the number of states to be maintained in memory can be kept small, at a cost of some loss of precision. Importantly, the FAU method can also speed up the analysis of very large finite models.

Fast adaptive uniformisation was implemented [6] and applied successfully on a variety of biological systems [6,17], but for transient probabilities only. Many useful quantitative analyses involve the computation of expected rewards, which can be *instantaneous* (incurred at time t) or *cumulated* (until time t). An example of an instantaneous reward property is the number of molecules of a given species at time $100s$, and of a cumulative property the expected number of reactions that occurred for the duration of $100s$. Although one can express cumulative reward properties by adding additional species to the model, for example by increasing the reward by 1 every time a reaction occurs, this has the disadvantage of introducing an additional dimension into the model and, as we show later, can severely affect the performance, resulting in higher memory requirement and a consequent loss of precision.

In this paper, we extend fast adaptive uniformisation for CTMCs to allow for efficient computation of cumulative reward properties, thus avoiding the overhead of adding the additional dimension. We cast our results in the framework of propagation models of [17]. We implement the method, including the reward extension, and integrate it into the probabilistic model checker PRISM. To show the practical applicability of FAU, we have applied it to a range of case studies from biology, demonstrating superior performance compared to existing techniques.

Related work. FAU generalises adaptive uniformisation [18] by accelerating the discretisation and neglecting states with insignificant probability. Standard uniformisation is implemented in a number of tools, including PRISM, which we enhance with the FAU functionality in this paper. SABRE [6] is the first tool to implement FAU without cumulative rewards. Both PRISM and SABRE support models written in Systems Biology Markup Language (SBML) as input, in addition to their native modelling languages. SABRE is a stand-alone tool available for download or as a web interface; it additionally offers deterministic approximations using differential equations (by the Runge-Kutta fourth order method), which is faster and leads to accurate results for large numbers of molecules. PRISM does not support deterministic approximations, but provides extensive support for temporal logic model checking that is appropriate for molecular networks where some species occur in small numbers, or the encoding of spatial information is needed, as we consider in this paper. PRISM also provides statistical model checking and Gillespie simulation. The tool MARCIE [19] implements FAU but does not support cumulative rewards. Further tools that support reward properties but not FAU include, for instance, MÖBIUS [4] and MRMC [11].

2 Preliminaries

We begin by giving an overview of the main definitions and results based on [13,6,17]. A *continuous-time Markov chain (CTMC)* is given by a set of discrete states S and the transition rate matrix $\mathcal{R}\colon S \times S \to \mathbb{R}_{\geq 0}$ where $\mathcal{R}(s,s) = 0$ for all $s \in S$. The rate $\mathcal{R}(s,s')$ determines the delay before the transition can occur, i.e. the probability of this transition being triggered within t time-units is $1 - e^{-\mathcal{R}(s,s')\cdot t}$. Let $E(s) \stackrel{\text{def}}{=} \sum_{s' \in S} \mathcal{R}(s,s')$ be the exit rate and define the generator matrix Q by $Q \stackrel{\text{def}}{=} R - \mathit{diag}(E)$, where $\mathit{diag}(E)$ is the $S \times S$ matrix with E on its diagonal and zero everywhere else. Then $\pi_t\colon S \to \mathbb{R}_{\geq 0}$, the transient probability vector at time t, can be expressed as $\pi_t = \pi \cdot e^{Qt}$ given the initial probability vector π.

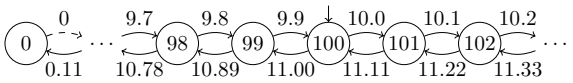

Fig. 1: Birth-death process.

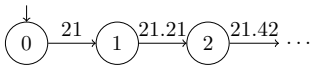

Fig. 2: Birth process.

We cast our method in the framework of *continuous-time (linear) propagation models* [17, Section 2.3.3] which generalise continuous-time Markov chains. We now recall the relevant results from [17].

Definition 1 (Continuous-time propagation model). *A* continuous-time propagation model (CTPM) *is a tuple* $\mathcal{M} = (S, \pi, \mathcal{R})$*, where*

- S *is a countable or finite set of* states*,*
- $\pi\colon S \to \mathbb{R}_{\geq 0}$ *where* $|\{s \in S \mid \pi(s) > 0\}| < \infty$ *is an* initialisation vector*, and*
- $\mathcal{R}\colon S \times S \to \mathbb{R}_{\geq 0}$ *is a* transition matrix*, such that for all* $s \in S$ *we have* $|\{s' \in S \mid \mathcal{R}(s, s') > 0\}| < \infty$*.*

The transition matrix $\mathcal{R}$ assigns a rate $\mathcal{R}(s, s')$ to each pair of states, as for CTMCs, and the initialisation vector π assigns an initial mass value $\pi(s)$ to each state $s \in S$. There are only finitely many states to which a positive mass is assigned initially. The models we consider are *finitely branching*, that is, for each state there are only finitely many states to which this state has a positive transition rate.

A CTPM is a CTMC if $\sum_{s \in S} \pi(s) = 1$ and $\mathcal{R}(s, s) = 0$ for all $s \in S$.

Example 1 (Continuous-time Markov chain). In Fig. 1, we depict a CTPM [7, 001-01], a so-called *birth-death process*. Each state s is a natural number describing the number s of molecules of a given species. In each state s, a new molecule can appear with rate $\lambda \cdot s$, and disappear with rate $\mu \cdot s$ for $\lambda \stackrel{\text{def}}{=} 0.1, \mu \stackrel{\text{def}}{=} 0.11$. We thus have $\mathcal{R}(s, s+1) \stackrel{\text{def}}{=} \lambda \cdot s$ for all $s \geq 0$, $\mathcal{R}(s, s-1) \stackrel{\text{def}}{=} \mu \cdot s$ for $s \geq 1$ and $\mathcal{R}(\cdot, \cdot) \stackrel{\text{def}}{=} 0$ otherwise. We assume that $\pi(100) \stackrel{\text{def}}{=} 1$ and for the other states $\pi(\cdot) \stackrel{\text{def}}{=} 0$. Thus, the model is a CTMC.

To reason about the timed behaviour of a CTPM, we now define its *generator matrix* which generalises that for CTMCs.

Definition 2 (Generator matrix). *The* generator matrix $Q(\mathcal{M})\colon S \times S \to \mathbb{R}$ *of a CTPM* $\mathcal{M}$ *is defined so that*

- $Q(\mathcal{M})(s, s') \stackrel{\text{def}}{=} \mathcal{R}(s, s')$ *for* $s, s' \in S$ *with* $s \neq s'$*, and*
- $Q(\mathcal{M})(s, s) \stackrel{\text{def}}{=} \mathcal{R}(s, s) - \sum_{\substack{s' \in S, \\ s' \neq s}} \mathcal{R}(s, s')$*.*

The *propagation process*, which propagates probability mass or expectation values, is then defined as follows. Note that, for CTMCs, $\pi_t(s)$ is the (transient) probability that the model resides in state s at time t.

Definition 3 (Propagation process). *Given a CTPM* $\mathcal{M} = (S, \pi, \mathcal{R})$*, the* propagation process at time t*,* $\pi_t(\mathcal{M})\colon S \to \mathbb{R}$*, is defined as the solution of the differential equation*

$$\dot{\pi}(s') \stackrel{\text{def}}{=} \sum_{s \in S} \pi(s) \cdot Q(\mathcal{M})(s, s')$$

at time t*, for* $s' \in S$*, given the initial value* π*.*

The *standard uniformisation* [10] method for CTMCs splits the CTMC into a discrete-time Markov chain (DTMC) and a Poisson process as follows. Define the DTMC P by $P \stackrel{\text{def}}{=} I + \frac{1}{\Lambda} \cdot Q$ where Λ is the uniformisation rate such that $\Lambda \geq \max_{s \in S} E(s)$. Then π_t can be computed as $\sum_{n=0}^{\infty} \pi_t(\Psi^{\Lambda})(n) \cdot \tau_n$ where $\pi_t(\Psi^{\Lambda})(n)$ is the value of the Poisson distribution with rate $\Lambda \cdot t$ at point n, and $\tau_n = \tau_{n-1} \cdot P$ for $n > 0$, $\tau_0 = \pi_0$. For a given precision ϵ, the summation can be truncated using the Fox and Glynn method [8].

The *fast adaptive uniformisation (FAU)* [6,17] is a variant of the *adaptive uniformisation* [18] which splits the CTMC into a DTMC and a birth process. For an infinite sequence $\mathbf{\Lambda} = (\Lambda_0, \Lambda_1, \ldots)$ of rates with $\Lambda_n \in \mathbb{R}_{\geq 0}$ for all $n \in \mathbb{N}$, the *birth process* is defined as the CTMC $\Phi^{\mathbf{\Lambda}} \stackrel{\text{def}}{=} (S, \pi, \mathcal{R})$, where

- $S \stackrel{\text{def}}{=} \mathbb{N}$,
- $\pi(0) \stackrel{\text{def}}{=} 1$ and $\pi(\cdot) \stackrel{\text{def}}{=} 0$ otherwise, and
- $\mathcal{R}(n, n+1) \stackrel{\text{def}}{=} \Lambda_n$ for $n \in \mathbb{N}$ and $\mathcal{R}(\cdot, \cdot) \stackrel{\text{def}}{=} 0$ otherwise.

Note that the Poisson process is a special case of the birth process with constant rates $\mathbf{\Lambda} = (\Lambda, \Lambda, \ldots)$.

Transient probabilities of birth processes can be approximated efficiently using specialised techniques [17, Section 4.3.2, Solution of the birth process]. This is possible by applying *standard uniformisation* [10] in a way which takes advantage of the particular structure of the process. Finally, transient probabilities of general CTPMs can be computed as follows, where we reformulate P_n in terms of the rate matrix, rather than the generator matrix used in [17].

Theorem 1 (Solving propagation models using a birth process). *Consider*

- *a CTPM* $\mathcal{M} = (S, \pi, \mathcal{R})$,
- *an infinite sequence of subsets* $\mathbf{S} = (S_0, S_1, \ldots)$ *with* $S_n \subseteq S$ *denoting* active states,
- *an infinite sequence* $\mathbf{\Lambda} = (\Lambda_0, \Lambda_1, \ldots)$ *with* $\Lambda_n \geq \sup_{s \in S_n} \sum_{\substack{s' \in S, \\ s' \neq s}} \mathcal{R}(s, s')$ *of* uniformisation rates,
- probability matrices $P_n(\mathcal{M})\colon S \times S \to \mathbb{R}_{\geq 0}$ *for* $n \in \mathbb{N}$, *where for* $s, s' \in S$ *we have*

$$P_n(\mathcal{M})(s, s') \stackrel{\text{def}}{=} \begin{cases} \frac{\mathcal{R}(s,s')}{\Lambda_n} & \text{if } s \neq s', \text{ and} \\ \frac{\mathcal{R}(s,s')}{\Lambda_n} + 1 - \sum_{\substack{s'' \in S, \\ s'' \neq s}} \frac{\mathcal{R}(s,s'')}{\Lambda_n} & \text{otherwise,} \end{cases}$$

- discrete-time distributions $\tau_n(\mathcal{M})\colon S \to \mathbb{R}_{\geq 0}$ *for* $n \in \mathbb{N}$ *with*

$$\tau_n(\mathcal{M})(s') \stackrel{\text{def}}{=} \begin{cases} \pi(s') & \text{if } n = 0, \text{ and} \\ \sum_{s \in S} \tau_{n-1}(\mathcal{M})(s) \cdot P_{n-1}(s, s') & \text{otherwise.} \end{cases}$$

We further require that $\{s \in S \mid \tau_n(\mathcal{M})(s) > 0\} \subseteq S_n$ *for* $n \in \mathbb{N}$.

Then we have that, at time t, *for each* $s \in S$*:*

$$\pi_t(\mathcal{M})(s) = \sum_{n=0}^{\infty} \pi_t(\Phi^{\mathbf{\Lambda}})(n) \cdot \tau_n(\mathcal{M})(s).$$

The *fast adaptive uniformisation* method [17] builds on the result above and works as follows. Starting with the initial distribution at step $n = 0$, at each step n the FAU explores a subset $\hat{S}_n$ of the states S_n. The sets $\hat{S}_n$ are constructed by taking $\hat{S}_{n-1}$, adding

the successor states $\{s' \in S \mid \exists s \in \hat{S}_{n-1}.\ \mathcal{R}(s,s') > 0\}$ of this set, and discarding states s with $\tau_n(\mathcal{M})(s) < \delta$, where δ is a fixed precision threshold. This process is repeated until step m, for instance so that $(1 - \sum_{n=0}^{m} \pi_t(\Phi^{\mathbf{\Lambda}})(n)) < \varepsilon$. Thus, we add the probability from the birth process at each step, and stop the state space exploration as soon as the value obtained this way is at least $1-\varepsilon$. In contrast to standard uniformisation, where the Fox and Glynn [8] algorithm can be utilised, we do not have an a priori step bound, but are still able to decide in a straightforward way when the infinite sum can be safely truncated.

Definition 4 (Fast Adaptive Uniformisation). *Let $\mathcal{M}$, $\mathbf{S} = (S_0, S_1, \ldots)$, and $\mathbf{\Lambda} = (\Lambda_0, \Lambda_1, \ldots)$ be as in Theorem 1. Further, consider*

- *a* truncation point $m \in \mathbb{N}$,
- *a finite sequence of subsets $\hat{\mathbf{S}} = (\hat{S}_0, \ldots, \hat{S}_m)$ with $\hat{S}_n \subseteq S_n$ denoting* active states *for $n \in \{1, \ldots, m\}$,*
- probability matrices $\hat{P}_n(\mathcal{M})\colon S \times S \to \mathbb{R}_{\geq 0}$ *for $n \in \{0, \ldots, m\}$ where*

$$\hat{P}_n(\mathcal{M})(s,s') \stackrel{\text{def}}{=} \begin{cases} P_n(\mathcal{M})(s,s') & \text{if } s \in \hat{S}_n, \text{ and} \\ 0 & \text{otherwise,} \end{cases}$$

- discrete-time distributions $\hat{\tau}_n(\mathcal{M})\colon S \to \mathbb{R}_{\geq 0}$ *for $n \in \mathbb{N}$ with*

$$\hat{\tau}_n(\mathcal{M})(s') \stackrel{\text{def}}{=} \begin{cases} \pi(s') & \text{if } n = 0, \text{ and} \\ \sum_{s \in S} \hat{\tau}_{n-1}(\mathcal{M}) \cdot \hat{P}_{n-1}(s,s') & \text{otherwise.} \end{cases}$$

We define the fast adaptive uniformisation (FAU) *value at time t for each $s \in S$ as*

$$\hat{\pi}_t(\mathcal{M}, \hat{\mathbf{S}}, \mathbf{\Lambda})(s) \stackrel{\text{def}}{=} \sum_{n=0}^{m} \pi_t(\Phi^{\mathbf{\Lambda}})(n) \cdot \tau_n(\mathcal{M})(s).$$

Example 2 (Fast Adaptive Uniformisation). We sketch how one can perform FAU for the CTMC from Example 1 according to Definition 4 and Theorem 1: only for state $s = 100$ the initial distribution is positive, so we can use $S_0 \stackrel{\text{def}}{=} \{100\}$. Then, we use $S_n \stackrel{\text{def}}{=} \{\max\{0, 100-n\}, \ldots, n\}$ and $\Lambda_n \stackrel{\text{def}}{=} (\lambda + \mu) \cdot n$. The corresponding birth process is sketched in Fig. 2. In Fig. 3, we depict S_n together with the relevant parts of the matrices P_n for $n = 0, 1, 2, \ldots, \infty$ (rounding-off the numbers). In addition, for $t = 0.1$ we provide the transient probabilities of being in the nth state of the birth process and the first n summands of $\pi_{0.1}(\mathcal{M})(100)$.

State $s = 0$ is absorbing, that is, once entered it cannot be left, and rates leading to a decrease in molecule count are higher than those leading to an increase. Thus, in the last line ("Step ∞") we see that, as n increases, the probability concentrates on the state $s = 0$. Thus, we can discard states with a high number of molecules from the reduced state sets $\hat{S}_n$, while retaining a sufficient amount of the total probability.

We now define instantaneous rewards, which can be used to express expected reward properties incurred at a given time. We annotate the models with state rewards.

Definition 5 (State reward structure). *A* state reward structure *for a CTMC $\mathcal{M} = (S, \pi, \mathcal{R})$ is a function* $\mathbf{r}\colon S \to \mathbb{R}_{\geq 0}$.

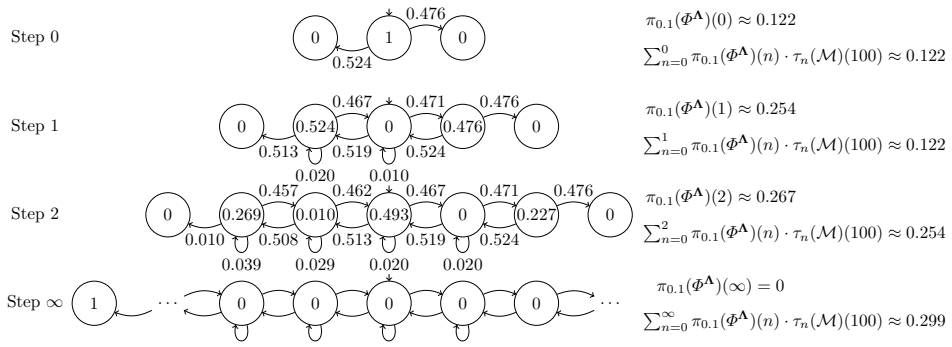

Fig. 3: Demonstration of Fast Adaptive Uniformisation.

Definition 6 (Instantaneous rewards). *Consider a CTMC $\mathcal{M} = (S, \pi, \mathcal{R})$ with state reward structure* $\mathbf{r}\colon S \to \mathbb{R}_{\geq 0}$ *and a time point $t \in \mathbb{R}_{\geq 0}$. The* instantaneous reward *is defined as*

$$\mathcal{I}_t(\mathcal{M}, \mathbf{r}) \stackrel{\text{def}}{=} \sum_{s \in S} \pi_t(\mathcal{M})(s) \cdot \mathbf{r}(s).$$

We show that instantaneous rewards can be easily accommodated within the FAU method, and we can approximate the expected mass value by terminating the state-space exploration using a criterion similar to the probability mass calculation in [17].

Definition 7 (Instantaneous reward approximation). *Let $\mathcal{M}$, $\mathbf{S} = (S_0, S_1, \ldots)$, and $\mathbf{\Lambda} = (\Lambda_0, \Lambda_1, \ldots)$ be as in Theorem 1 and let $\hat{\mathbf{S}}$ be as in Definition 4. Then we define*

$$\mathcal{I}_t(\mathcal{M}, \mathbf{r}, \hat{\mathbf{S}}, \mathbf{\Lambda}) \stackrel{\text{def}}{=} \sum_{s \in S} \hat{\pi}_t(\mathcal{M}, \mathbf{S}, \mathbf{\Lambda})(s) \cdot \mathbf{r}(s).$$

Corollary 1 (Error bounds for FAU). *Consider a CTMC $\mathcal{M} = (S, \pi, \mathcal{R})$ for which we have the uniformisation rates $\mathbf{\Lambda}$ (cf. Theorem 1) and a state reward structure* $\mathbf{r}$*. Consider $m \in \mathbb{N}$, $\hat{\mathbf{S}}$, and $\hat{\tau}$ as in Definition 4. Set $\mathbf{\Lambda} \stackrel{\text{def}}{=} (\Lambda, \Lambda, \ldots)$. Then if*

$$\max_{s \in S} \mathbf{r}(s) \cdot \left(1 - \sum_{n=0}^{m} \pi_t(\Phi^{\mathbf{\Lambda}})(n)\right) < \frac{\epsilon}{2} \text{ and } \max_{s \in S} \mathbf{r}(s) \cdot \left(1 - \sum_{s \in \hat{S}_m} \hat{\tau}_m(s)\right) < \frac{\epsilon}{2}$$

it follows that

$$\mathcal{I}_t(\mathcal{M}, \mathbf{r}) - \mathcal{I}_t(\mathcal{M}, \mathbf{r}, \hat{\mathbf{S}}, \mathbf{\Lambda}) < \epsilon.$$

Proof. Part of the expected reward value is lost due to the approximation of the infinite sum. This is accounted for by first inequality. By discarding states while exploring the state space, we lose further mass. This is accounted for by the second inequality. Adding up the maxima of the two errors, we can bound the error. □

Example 3 (FAU for Instantaneous Rewards). Consider the CTMC from Example 1 with a reward structure $\mathbf{r}$ assigning to each state s the number of molecules s. When using the transient probability values computed in Example 2, for the computation of the exact values we have $\mathcal{I}_{0.1}(\mathcal{M}, \mathbf{r}) = \sum_{s \in S} \hat{\pi}_t(\mathcal{M}, \mathbf{S}, \mathbf{\Lambda})(s) \cdot \mathbf{r}(s) \approx \ldots + 100 \cdot 0.299 + \ldots \approx 99.900$.

3 Cumulative Rewards

In this section, we extend the FAU method to reason about properties of the behaviour of a CTMC model cumulating the rewards *until* a given point of time. The correctness of the method is proved using the framework of CTPMs [17], where cumulative rewards were not considered.

For a given CTMC, we first extend its state space by adding *time-accumulating* states to remember how much time was spent in a specific state, and then, noting that the time-extended CTPM is not a CTMC, show how the expected reward computation can be approximated.

Definition 8 (Time-extended CTPM). *Given a CTMC* $\mathcal{M} = (S, \pi, \mathcal{R})$*, the* time-extended CTPM *is defined as*

$$\mathrm{ext}(\mathcal{M}) \stackrel{\text{def}}{=} (S_{\text{ext}}, \pi_{\text{ext}}, \mathcal{R}_{\text{ext}}), \text{ where}$$

- $S_{\text{ext}} \stackrel{\text{def}}{=} S \uplus S_{\text{acc}}$*, where for each* $s \in S$ *we have exactly one corresponding* time-accumulating $s_{\text{acc}} \in S_{\text{acc}}$*, that is,* $S_{\text{acc}} \stackrel{\text{def}}{=} \{s_{\text{acc}} \mid s \in S\}$*,*
- $\pi_{\text{ext}}(s) \stackrel{\text{def}}{=} \pi$ *for* $s \in S$ *and* $\pi_{\text{ext}}(\cdot) \stackrel{\text{def}}{=} 0$ *otherwise, and*
- *the transition matrix* $\mathcal{R}_{\text{ext}} \colon S_{\text{ext}} \times S_{\text{ext}} \to \mathbb{R}_{\geq 0}$ *is defined such that, for* $s_{\text{ext}}, s'_{\text{ext}} \in S_{\text{ext}}$*, we have*

$$\mathcal{R}_{\text{ext}}(s_{\text{ext}}, s'_{\text{ext}}) \stackrel{\text{def}}{=} \begin{cases} \mathcal{R}(s_{\text{ext}}, s'_{\text{ext}}) & \text{if } s_{\text{ext}}, s'_{\text{ext}} \in S \text{ and } s_{\text{ext}} \neq s'_{\text{ext}}, \\ 1 & \text{if } s_{\text{ext}} = s \in S \text{ and } s'_{\text{ext}} \in \{s_{\text{ext}}, s_{\text{acc}}\}, \\ 0 & \text{otherwise.} \end{cases}$$

We now use time-extended CTPMs to prove the central theorem of the paper. This is achieved by first approximating the residence time, and then the cumulative reward, by considering the reward per time unit of residing in a given state. We use the *mixed birth process probability* $\Psi^{\mathbf{\Lambda}}(n) = \frac{1}{\Lambda_n} \cdot \sum_{i=n+1}^{\infty} \pi_t(\Phi^{\mathbf{\Lambda}})(i)$, which denotes the probability that more than n state changes happen within time t in the birth process, divided by the nth uniformisation rate. This is used to collect the time *spent* in given state, as opposed to the probability $\pi_t(\Phi^{\mathbf{\Lambda}})(i)$ *to be in* a state at a given point of time.

Theorem 2 (Residence time). *Consider a CTMC* $\mathcal{M} = (S, \pi, \mathcal{R})$ *and let* $\rho_t(\mathcal{M}) \colon S \to \mathbb{R}_{\geq 0}$ *be defined as*

$$\rho_t(\mathcal{M})(s) \stackrel{\text{def}}{=} \int_0^t \pi_u(s)\,\mathrm{d}u$$

for $s \in S$*. Then we have*

$$\rho_t(\mathcal{M})(s) = \sum_{n=0}^{\infty} \Psi^{\mathbf{\Lambda}}(n) \cdot \tau_n(\mathcal{M})(s),$$

for $s \in S$*,* τ *and* $\mathbf{\Lambda}$ *as in Theorem 1 and* $\Psi^{\mathbf{\Lambda}}(n) \stackrel{\text{def}}{=} \frac{1}{\Lambda_n} \cdot \sum_{i=n+1}^{\infty} \pi_t(\Phi^{\mathbf{\Lambda}})(i)$*.*

Proof. Assume $\mathrm{ext}(\mathcal{M}) = (S_{\text{ext}}, \pi_{\text{ext}}, \mathcal{R}_{\text{ext}})$. Then, by definition of the structure of the time-extended CTPM, we have for $s \in S$ that

$$\pi_t(\mathcal{M})(s) = \pi_t(\mathcal{M}_{\text{ext}})(s), \text{ and } \rho_t(\mathcal{M})(s) = \pi_t(\mathcal{M}_{\text{ext}})(s_{\text{acc}}).$$

By the structure of the time-extended CTPM, we have

$$\tau_0(\mathcal{M}_{\text{ext}})(s_{\text{acc}}) = 0,$$
$$\tau_{n+1}(\mathcal{M}_{\text{ext}})(s_{\text{acc}}) = \frac{1}{\Lambda_n} \cdot \tau_n(\mathcal{M}_{\text{ext}})(s) + \tau_n(\mathcal{M}_{\text{ext}})(s_{\text{acc}}) = \frac{1}{\Lambda_n} \cdot \tau_n(\mathcal{M})(s) + \tau_n(\mathcal{M}_{\text{ext}})(s_{\text{acc}})$$

and thus

$$\tau_n(\mathcal{M}_{\text{ext}})(s_{\text{acc}}) = \sum_{i=0}^{n-1} \frac{1}{\Lambda_i} \cdot \tau_i(\mathcal{M})(s).$$

From this and by Theorem 1 we have

$$\begin{aligned}
\pi_t(\mathcal{M}_{\text{ext}})(s_{\text{acc}}) &= \sum_{n=0}^{\infty} \pi_t(\Phi^{\mathbf{\Lambda}})(n) \cdot \tau_n(\mathcal{M}_{\text{ext}})(s_{\text{acc}}) \\
= \sum_{n=0}^{\infty} \pi_t(\Phi^{\mathbf{\Lambda}})(n) \cdot \sum_{i=0}^{n-1} \frac{1}{\Lambda_i} \cdot \tau_i(\mathcal{M})(s) &= \sum_{n=0}^{\infty} \sum_{i=0}^{n-1} \frac{1}{\Lambda_i} \cdot \tau_i(\mathcal{M})(s) \cdot \pi_t(\Phi^{\mathbf{\Lambda}})(n) \\
= \sum_{i=0}^{\infty} \sum_{n=i+1}^{\infty} \frac{1}{\Lambda_i} \cdot \tau_i(\mathcal{M})(s) \cdot \pi_t(\Phi^{\mathbf{\Lambda}})(n) &= \sum_{i=0}^{\infty} \left(\frac{1}{\Lambda_i} \cdot \sum_{n=i+1}^{\infty} \pi_t(\Phi^{\mathbf{\Lambda}})(n) \right) \cdot \tau_i(\mathcal{M})(s) \\
&= \sum_{i=0}^{\infty} \Psi^{\mathbf{\Lambda}}(i) \cdot \tau_i(\mathcal{M})(s) = \sum_{n=0}^{\infty} \Psi^{\mathbf{\Lambda}}(n) \cdot \tau_n(\mathcal{M})(s). \qquad \square
\end{aligned}$$

The above theorem splits the behaviour of a CTMC into the birth process and a discrete-time process that determines the time spent in specific states of the CTMC. Thus, we can now apply the FAU to compute cumulative reward properties. To do this, we accumulate rewards for being in a state over time. Transition rewards $\mathbf{r}_t \colon S \times S \to \mathbb{R}_{\geq 0}$ are obtained for moving from one state to another.

We do not explicitly consider transition rewards for CTMCs here. However, given state rewards $\mathbf{r}$ and transition rewards $\mathbf{r}_t$, we can define cumulative reward rates $\mathbf{r}'$ as $\mathbf{r}'(s) \stackrel{\text{def}}{=} \mathbf{r}(s) + \sum_{s' \in S} \mathcal{R}(s, s') \cdot \mathbf{r}_t(s, s')$. For the properties under consideration, this new reward structure is equivalent to using transition rewards, as shown in [12, Equation 6].

We stress that time-extended CTPMs are used here only in the proof, and never constructed in our method.

Definition 9 (Cumulative rewards). *Consider a CTMC $\mathcal{M} = (S, \pi, \mathcal{R})$ with state reward structure $\mathbf{r}\colon S \to \mathbb{R}_{\geq 0}$ and a time duration $t \in \mathbb{R}_{\geq 0}$. The* cumulative reward value *is defined as*

$$\mathcal{C}_t(\mathcal{M}, \mathbf{r}) \stackrel{\text{def}}{=} \int_0^t \sum_{s \in S} \pi_u(\mathcal{M})(s) \cdot \mathbf{r}(s) \, \mathrm{d}u.$$

We now use the results from Theorem 2 to compute the reward obtained until a given point of time. The following corollary follows directly from Theorem 2 and can be used to approximate cumulative rewards.

Corollary 2 (Computing rewards). *For a CTMC $\mathcal{M} = (S, \pi, \mathcal{R})$ with state reward structure $\mathbf{r}\colon S \to \mathbb{R}_{\geq 0}$ and a time duration $t \in \mathbb{R}_{\geq 0}$, we have*

$$\mathcal{C}_t(\mathcal{M}, \mathbf{r}) = \sum_{s \in S} \rho_t(\mathcal{M})(s) \cdot \mathbf{r}(s) = \sum_{n=0}^{\infty} \sum_{s \in S} \Psi^{\mathbf{\Lambda}}(n) \cdot \tau_n(\mathcal{M})(s) \cdot \mathbf{r}(s).$$

Definition 10 (Cumulative reward approximation). *Let $\mathcal{M}$, $\mathbf{S} = (S_0, S_1, \ldots)$, and $\mathbf{\Lambda} = (\Lambda_0, \Lambda_1, \ldots)$ be as in Theorem 1 and let $m \in \mathbb{N}$ and $\hat{\mathbf{S}}$ be as in Definition 4. Then we define*

$$\mathcal{C}_t(\mathcal{M}, \mathbf{r}, t, \hat{\mathbf{S}}, \mathbf{\Lambda}) \stackrel{\text{def}}{=} \sum_{n=0}^{m} \sum_{s \in S} \Psi^{\mathbf{\Lambda}}(n) \cdot \hat{\tau}_n(\mathcal{M})(s) \cdot \mathbf{r}(s).$$

Calculating the cumulative rewards is of similar complexity to calculating the instantaneous rewards. After each step n, we multiply the probability in the discrete-time process by the corresponding cumulative reward and the value from Ψ, and then sum up the values obtained this way. The time overhead to compute the accumulated reward values is negligible. More importantly, it is not necessary to extend the state space, and hence the space complexity compared to FAU is not increased.

The corollary can be seen as a generalisation of a previous result [12, Theorem 1], where the computation of cumulative reward-based properties is also considered. However, the analysis in [12] relies on complete exploration of the state space and uses the special case $\mathbf{\Lambda} = (\Lambda, \Lambda, \ldots)$, which reduces the birth process to a Poisson process.

The computation of the error and the bounds on the number of steps is more involved for cumulative rewards than for instantaneous rewards, as shown in Corollary 1. The precision which can be achieved depends on the structure of the CTMC and the state rewards. We often have models in which, for each state, the sum of the rates to new states (further away from initial states) is bounded. We remark that this does not restrict the rates back to previously visited states. For this class of models, which includes many realistic examples as shown below, we derive error bounds as follows.

Corollary 3 (Error bound cumulative). *Consider a CTMC $\mathcal{M} = (S, \pi, \mathcal{R})$ for which we have a fixed Λ so that for each $n \in \mathbb{N}$ and $s \in S_n$ (cf. Theorem 1) we have that $\sum_{s' \in S_{n+1}} \mathcal{R}(s, s') \leq \Lambda$. Further, consider a state reward structure $\mathbf{r}$ so that we have fixed constants $c, d \in \mathbb{R}_{\geq 0}$ where for all $n \in \mathbb{N}$ and $s \in S_n$ we have $\mathbf{r}(s) \leq c + dn$. Consider $m \in \mathbb{N}$, $\hat{\mathbf{S}}$, and $\hat{\tau}$ as in Definition 4. Set $\mathbf{\Lambda}_b \stackrel{\text{def}}{=} (\Lambda, \Lambda, \ldots)$ and $B \stackrel{\text{def}}{=} t \cdot (c + d + d\Lambda t)$. If*

$$B - \sum_{n=0}^{m} (c + dn) \cdot \Psi^{\mathbf{\Lambda}_b}(n) < \frac{\epsilon}{2} \text{ and } B \cdot \left(1 - \sum_{s \in \hat{S}_m} \hat{\tau}_m(s)\right) < \frac{\epsilon}{2}$$

then we have

$$\mathcal{C}_t(\mathcal{M}, \mathbf{r}) - \mathcal{C}_t(\mathcal{M}, \mathbf{r}, \hat{\mathbf{S}}, \mathbf{\Lambda}) < \epsilon.$$

Proof. When applying the FAU method, the worst case of reward loss is when we have the birth process $\Phi^{\mathbf{\Lambda}_b} = (\mathbb{N}, \pi, \mathcal{R})$ with reward structure $\mathbf{r}$, so that, for all $n \in \mathbb{N}$, we have $\mathbf{r}(n) \stackrel{\text{def}}{=} c + dn$. Denote the total accumulated reward until time t for this model by B. Thus, we lose no more reward than $B - \sum_{n=0}^{m} (c + dn) \cdot \Psi^{\mathbf{\Lambda}_b}(n)$ in case we use $\hat{S}_n = S_n$ and perform m steps in the FAU.

To take into account the loss of rewards from using $\hat{S}_n \subseteq S_n$, we consider the total probability lost $(1 - \sum_{s \in \hat{S}_m} \hat{\tau}_m(s))$. In the worst case, this probability was already lost at the beginning. In this case, we lose up to $B \cdot (1 - \sum_{s \in \hat{S}_m} \hat{\tau}_m(s))$.

By adding up the two sources of error, we obtain the result. □

If the rates or rewards are increasing more quickly, e.g., if we have a quadratic increase in the rewards, that is, $\mathbf{r}(s) \leq c + dn^2$ for $s \in \hat{S}_n$, the bounds on the error can be obtained

using similar reasoning for a different value of B. Because of the simple structure of birth processes, it is possible to quickly approximate $\sum_{n=0}^{m}(c+dn)\cdot\Psi^{\mathbf{\Lambda}}(n)$ to find the value m to terminate the approximation in the worst case.

Example 4 (FAU for Cumulative Rewards). We reconsider the CTMC from Example 1 for which we computed transient probabilities in Example 2. We are interested in the expected total number of changes to the number of molecules, and thus assign a reward of 1 to each state change. As discussed, we transform these transition rewards into a state reward structure $\mathbf{r}$. For instance, state $s = 100$ has two transitions with rates 10 and 11, both with reward of 1, so that the state reward here is $10 \cdot 1 + 11 \cdot 1 = 21$. We have $\Psi^{\mathbf{\Lambda}}(0) \approx 0.042$, $\Psi^{\mathbf{\Lambda}}(1) \approx 0.029$, $\Psi^{\mathbf{\Lambda}}(2) \approx 0.017$. To compute cumulative rewards, according to Corollary 2 we can proceed as follows: after each step n of Example 2 and Fig. 3, for each $s \in S_n$ ($s \in \hat{S}_n$) we compute the product $\Psi^{\mathbf{\Lambda}}(n) \cdot \tau_n(s) \cdot \mathbf{r}(s)$ and build the sum $v(n) = \sum_{s\in S} \Psi^{\mathbf{\Lambda}}(n) \cdot \hat{\tau}_n(\mathcal{M})(s) \cdot \mathbf{r}(s)$ of these values. States $s \in S \setminus S_n$ need not be considered, because for those $\tau_n(s) = 0$. This value $v(n)$ is then added to the partially computed total cumulative reward. In the example, we have $v(0) \approx 0.042 \cdot 1 \cdot 100$, $v(1) \approx 0.029 \cdot (0.524 \cdot 99 + 0.476 \cdot 101)$, $v(2) = 0.017 \cdot (0.269 \cdot 98 + 0.010 \cdot 99 + 0.493 \cdot 100 + 0.227 \cdot 102)$. Finally, we obtain $\mathcal{C}_{0.1}(\mathcal{M}, \mathbf{r}) \approx 2.099$.

4 Case Studies and Implementation

We have integrated the fast adaptive uniformisation method in the probabilistic model checker PRISM [14] and intend to make it available in one of the next PRISM releases. Our implementation builds on top of the "explicit" engine, and is written in Java. Models can be input in the native language of PRISM or SBML. Properties are specified as non-nested continuous stochastic logic (CSL) [1] formulae extended with the reward operator [13], as either time-bounded until, or instantaneous or cumulative reward properties.

To show the practical applicability of our method, we apply it to three case studies. We terminate the state space exploration once we obtain $(1 - \sum_{n=0}^{m} \pi_t(\Phi^{\mathbf{\Lambda}}(n))) < \varepsilon$ for an adequate ε, and discard states with probability of less than δ in the discrete-time process. Experiments were performed on a Linux computer with an Intel i7-3770 processor with 3.40GHz and 32GB of RAM.

Wherever possible, we have compared our results to the PRISM engine which performs best for that particular model. This includes comparison with the symbolic engines of PRISM ("mtbdd" and "hybrid"), which tended to perform worse than the "explicit" engine on our examples, likely due to loss of regularity. We note that symbolic engines cannot handle infinite-state models employed here, but the "explicit" engine is able to, provided that the reachable state space is finite. Conventional methods could perform better than FAU when the state space is sufficiently small, in view of the additional overhead necessary for FAU. We anticipate that the performance of PRISM is indicative of modern probabilistic model checkers, and therefore our conclusions are more generally applicable.

In this paper, we do not compare against simulation-based approaches, such as approximate probabilistic model checking available in PRISM (probability estimation and statistical hypothesis testing); while simulation has the advantage of not requiring the generator matrix to be constructed, and hence does not suffer from state-space explosion, it is sensitive to the size of time bounds and can only guarantee error bounds with a given confidence interval. FAU can provide guarantees for an arbitrary precision by controlling δ, although reducing δ will generally incur higher memory requirements. Investigating the trade-off between FAU and simulation-based techniques deserves further study.

Model	Time (s)	States	Lost	Molecules	Reactions
001-01	1	321	8.7060E-09	60.6531	826.2856
001-03	2	347	9.6165E-09	0.6738	2,085.8503
001-04	1	163	8.1621E-09	6.0653	82.6286
001-05	22	2,999	1.1078E-08	6,065.3065	82,628.5611
001-06	1	321	8.7060E-09	60.6531	826.2856
001-07	51	161,617	2.0779E-08	60.6531	826.2856
001-08	2	321	8.7060E-09	60.6531	826.2856
001-18	1	277	8.5217E-09	77.8801	464.5184
001-19	1	321	8.7060E-09	60.6531	826.2856
002-01	2	44	5.9691E-09	9.9326	90.0674
002-02	2	151	7.7930E-09	99.3262	900.6738
002-03	2	107	8.3948E-09	49.6631	450.3369
002-04	19	1,377	1.0983E-08	9,932.6204	90,067.3790
002-05	1	151	7.7930E-09	99.3262	900.6738
002-06	5	36,255	9.9788E-09	99.3262	900.6738
002-07	1	151	7.7930E-09	99.3262	900.6738
002-08	2	44	5.9691E-09	9.9326	90.0674
003-01	2	48	5.3131E-09	28.5423	64.7560
003-02	2	156	7.5890E-09	144.9960	573.9888
003-05	2	48	5.3131E-09	35.7289	64.7560
004-01	1	124	8.0978E-09	24.9989	275.0011
004-02	1	173	9.0356E-09	25.0000	525.0000
004-03	4	773	3.5419E-08	25.0000	5,024.9999
ext. 001-01	66	161,617	2.0779E-08	60.6531	826.2856

Table 1: Discrete Stochastic Model Test Suite Results.

Note that the performance figures given in the tables reflect the relative speeds of engines at the time of writing, and can change due to further optimisation.

4.1 Discrete Stochastic Model Test Suite

The Discrete Stochastic Model Test Suite [7] is a test suite of models encoded in the Systems Biology Markup Language (SBML), for which values of certain properties have been computed up to a given precision. It is aimed at stochastic simulator developers who can evaluate the accuracy of their tools against known results.

We used PRISM's SBML import functionality[3] to convert SBML to PRISM files. The models have infinitely many states, and so cannot be handled by existing PRISM engines (except "explicit", providing the reachable state space is finite). As the SBML import does not yet support the SBML feature of *events*, we were only able to analyse 35 out of the 39 test models. For this case study, we chose $\varepsilon = 10^{-9}$ and $\delta = 10^{-13}$ and apply analyses for a time bound of 50, which is the largest one for which results are included in the SBML models. The results for a selection of the models are given in Table 1. For each "Model", we give the "Time (s)" in seconds needed to perform the analysis, the maximal number of "States" in memory, and the probability "Lost" through approximation. The column "Molecules" is an instantaneous reward property that returns the expected number of molecules of the first species of the model under consideration. The column "Reactions" is the expected number of reactions until time 50, which is a cumulative reward property. In the table, each row corresponds to two analyses; however, the computation time is the same for both since the same number of states had to be explored.

All analyses (with the exception of "ext. 001-01" not originally from the test suite, see below) took less than a minute. The results we obtain for "Molecules" agree with those provided by the test suite, for the number of decimal places given there (values for "Reactions" are not provided by the test suite). For the model "001-01", we attempted a naive approach to compute the number of reactions by adding a new species "Reactions", increasing the dimensionality. As can be seen from results given in the last row ("ext. 001-01") of Table 1, these performance figures were much worse than for our implementation. We remark that these figures are similar to those for the (unmodified) "001-07", in which also a species tracking a specific reaction is introduced.

[3] `http://www.prismmodelchecker.org/manual/RunningPRISM/SupportForSBML`

N	T	Fastest PRISM engine - explicit				FAU				
		Time (s)	States	Finished	Reactions	Time (s)	States	Lost	Finished	Reactions
1	10000	3	169	0.0593	15.3193	**2**	169	1.6882E-09	0.0593	15.3193
1	50000	**1**	169	0.9999	26.9997	33	169	1.7133E-09	0.9999	26.9997
1	100000	**2**	169	1.0000	27.0000	140	169	1.7892E-09	1.0000	27.0000
2	10000	**1**	5,748	0.0224	37.1224	7	**5,299**	1.3024E-08	0.0224	37.1224
2	50000	**2**	5,748	0.9999	51.2958	41	**5,299**	1.5028E-08	0.9999	51.2958
2	100000	**2**	5,748	1.0000	51.2963	155	**5,299**	1.5090E-08	1.0000	51.2963
3	10000	**15**	93,538	0.0138	59.7229	145	**67,292**	1.0059E-07	0.0138	59.7229
3	50000	**52**	93,538	0.9999	75.0530	179	**67,292**	1.0994E-07	0.9999	75.0530
3	100000	**96**	93,538	1.0000	75.0536	437	**67,292**	1.1002E-07	1.0000	75.0536
4	10000	**268**	970,539	0.0103	82.6250	835	**514,414**	5.6703E-07	0.0103	82.6250
4	50000	1,039	970,539	0.9998	98.6211	**872**	**514,414**	5.8001E-07	0.9998	98.6211
4	100000	1,976	970,539	1.0000	98.6218	**1,209**	**514,414**	5.8009E-07	1.0000	98.6218
5	10000	3,463	7,377,039	0.0085	105.6602	**2,370**	**2,814,235**	2.9759E-06	0.0085	105.6602
5	50000	-	-	-	-	**2,416**	**2,814,235**	2.9907E-06	0.9998	122.0891
5	100000	-	-	-	-	**2,815**	**2,814,235**	2.9907E-06	1.0000	122.0897
6	10000	-	-	-	-	**5,453**	**12,163,811**	1.3377E-05	0.0074	128.7586
6	50000	-	-	-	-	**5,644**	**12,163,811**	1.3393E-05	0.9998	145.4913
6	100000	-	-	-	-	**5,960**	**12,163,811**	1.3393E-05	1.0000	145.4920

Table 2: DNA Strand Displacement Results.

4.2 DNA Strand Displacement

DNA strand displacement (DSD) [20] is a mechanism for performing computation with DNA molecules. A variety of logic circuits can be designed and implemented using DSD. Initial species of DNA are mixed together in a reaction tube, and then strand displacement reactions proceed autonomously, relying solely on hybridisation between complementary nucleotide sequences to perform computational steps. In this case study, we consider transducer gates modelled and analysed in [15, Section 2] (example `transducer_K=3.sm`). This model features the parameter N, which corresponds to the number of copies for initial species, and K, the number of transducers placed in series.

We are interested in the probability that the computation is finished by time T ("Finished"), an instantaneous reward property, and the expected total number of reactions ("Reactions"), a cumulative reward property. We fix $K = 3$ and provide results for different N and T in Table 2. The state space of this case study is small enough to be compared against existing methods in PRISM. We included the results for the "explicit" engine because it was the fastest. In each row, the best performance in terms of state-space size or time is highlighted in boldface.

Note also that the FAU method is able to handle larger models than existing PRISM engines, and obtains better performance for larger model instances.

4.3 DNA Walkers

We consider models of *DNA walkers* [22], which can also be used to design logic circuits on the nanoscale. The main difference from DSD designs is that a DNA walker operates on a track of DNA strands (called anchorages) tethered to a surface, rather than in solution, and thus the model has to incorporate spatial information. An example of an XOR circuit is shown in Fig. 4. We assign True/False values to absorbing anchorages. The walker starts in the Initial position and can navigate down a series of junctions [21]. An enzyme cuts the anchorage when the walker is attached, allowing the walker to step onto the next anchor-

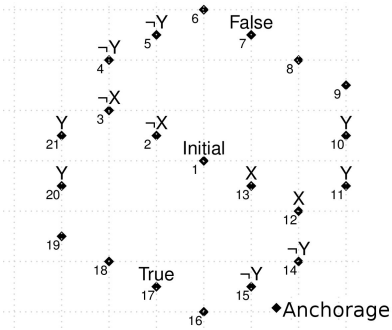

Fig. 4: Walker 'XOR' circuit. Adding the input X will block the anchorages labelled $\neg X$. Once the walker reaches True or False the computation ends.

Model	Time (s)	States	Lost	Signal	Steps	Blocked (s)
xor(X, Y)	4126	228,803	1.9736E-02	0.6455	7.7696	606.2731
xor($\neg X, Y$)	4070	228,803	1.9736E-02	0.6455	7.7696	606.2731
xor($X, \neg Y$)	4684	239,680	2.2587E-02	0.5979	7.5610	659.3715
xor-($\neg X, \neg Y$)	4593	239,680	2.2587E-02	0.5979	7.5610	659.3715
xor-S-(X, Y)	2970	215,544	1.6719E-02	0.5374	8.8363	133.1672
xor-S-($\neg X, Y$)	3027	215,544	1.6719E-02	0.5374	8.8363	133.1672
xor-S-($X \neg, Y$)	3651	233,063	1.8775E-02	0.5473	8.4049	146.7377
xor-S-($\neg X, \neg Y$)	3630	233,063	1.8775E-02	0.5473	8.4049	146.7377
xor-large-(X, Y)	18382	443,584	5.1855E-02	0.5661	9.5020	577.2680
xor-large-($\neg X, Y$)	18142	443,584	5.1855E-02	0.5661	9.5020	577.2680
xor-large-($X \neg, Y$)	19418	455,685	5.2995E-02	0.5674	9.4983	567.3420
xor-large-($\neg X, \neg Y$)	18114	455,685	5.2995E-02	0.5674	9.4983	567.3420

Table 3: DNA Walkers.

age. Depending on prior input, certain anchorages can be blocked, which in turn directs the walker at each junction.

A Markov chain model of the walker was developed [5] previously, and in this paper we apply model checking with FAU, using the parameter set $\varepsilon = 10^{-6}$ and $\delta = 10^{-8}$. We analyse three XOR-circuits, from Fig. 4 and two variants, summarising the results in Table 3. We model check the expected number of steps (column "Steps") and the probability of walkers reaching the desired anchorage (column "Signal") by time $T = 200$ min. The walker occasionally steps over blockades or between tracks, which may cause it to reach the wrong answer. Determining the size of the reachable state space appears to be a hard problem, not unlike determining the number self-avoiding walks on a lattice. We estimate the size to be around $1 \cdot 10^7$ and $9 \cdot 10^8$ reachable states for the normal and large tracks, respectively. This state space is too large to construct the models symbolically and compare against other PRISM engines.

The unmodified track, shown in Fig. 4, is "xor", and the suffix "-S" indicates that only one blocker is used instead of two consecutive ones, whereas suffix "-large" indicates a track with more anchorages. The expected number of steps correlates well with the track layout: when fewer anchorage are blocked ("-S"), the walker takes more steps on average. A larger track also results in more steps taken on average. Because the track has a point-symmetry, the results for inputs X, Y and $\neg X, Y$ are the same, as well as for inputs $\neg X, \neg\ Y$ and $X, \neg Y$. Occasionally, the blockade mechanism fails to block an anchorage. Column "Blocked" shows how much time the walker spends on anchorages that were supposed to be blocked, which is in line with expectations.

5 Conclusion

In this paper, we have extended fast adaptive uniformisation so that it can also be applied to cumulative reward properties. Cumulative measures allow one to express many important quantitative properties, such as the expected number of times a certain reaction happens and the average percentage of time the system spends in a given state. Our method does not introduce a significant overhead to the analysis, and in particular does not require the explicit construction of the extended state space of the underlying continuous-time propagation model. In contrast to simulation-based approaches, we can compute guaranteed error bounds for properties, as opposed to ensuring a given confidence interval. We have applied it to several case studies, obtaining superior performance in virtually all cases compared to existing methods.

Acknowledgements. The authors are supported in by the ERC Advanced Grant VERIWARE and a Microsoft Research PhD Studentship (FD). We would like to thank Taolue Chen and Andrew Phillips for useful discussion, and Chris Thachuk for his help in preparing the DSD models.

References

1. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.K.: Model-checking continuous-time Markov chains. ACM TCS 1(1), 162–170 (2000)
2. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Performance evaluation and model checking join forces. Commun. ACM 53(9), 76–85 (2010)
3. 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)
4. Clark, G., Courtney, T., Daly, D., Deavours, D., Derisavi, S., Doyle, J.M., Sanders, W.H., Webster, P.: The möbius modeling tool. In: PNPM. pp. 241–250 (2001)
5. Dannenberg, F., Kwiatkowska, M., Thachuk, C., Turberfield, A.: DNA walker circuits: computational potential, design, and verification (2013), Proc. DNA19, to appear
6. Didier, F., Henzinger, T.A., Mateescu, M., Wolf, V.: Sabre: A tool for stochastic analysis of biochemical reaction networks. In: QEST. pp. 193–194 (2010)
7. Evans, T.W., Gillespie, C.S., Wilkinson, D.J.: The SBML discrete stochastic models test suite. Bioinformatics 24(2), 285–286 (2008)
8. Fox, B.L., Glynn, P.W.: Computing Poisson probabilities. Comm. ACM 31(4), 440–445 (1988)
9. Heath, J., Kwiatkowska, M., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. Theoretical Computer Science 319(3), 239–257 (2008)
10. Jensen, A.: Markoff chains as an aid in the study of Markoff processes. Skand. Aktuarietidskr. 36, 87–91 (1953)
11. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. PEVA 68(2), 90–104 (2011)
12. Kwiatkowska, M., Norman, G., Pacheco, A.: Model checking expected time and expected reward formulae with random time bounds. CMA 51, 305–316 (2006)
13. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic model checking. In: SFM'07. LNCS (Tutorial Volume), vol. 4486, pp. 220–270. Springer (2007)
14. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. pp. 585–591 (2011)
15. Lakin, M., Parker, D., Cardelli, L., Kwiatkowska, M., Phillips, A.: Design and analysis of DNA strand displacement devices using probabilistic model checking. Journal of the Royal Society Interface 9(72), 1470–1485 (2012)
16. Mateescu, M., Wolf, V., Didier, F., Henzinger, T.A.: Fast adaptive uniformisation of the chemical master equation. IET Syst Biol 4(6), 441–52 (2010)
17. Mateescu, M.: Propagation Models for Biochemical Reaction Networks. Ph.D. thesis, EPFL (2011)
18. van Moorsel, A.P.A., Sanders, W.H.: Adaptive uniformization. ORSA Communications in Statistics: Stochastic Models 10(3), 619–648 (1994)
19. Schwarick, M., Heiner, M., Rohr, C.: MARCIE - model checking and reachability analysis done efficiently. In: QEST. pp. 91–100 (2011)
20. Seelig, G., Soloveichik, D., Zhang, D.Y., Winfree, E.: Enzyme-free nucleic acid logic circuits. Science 314(5805), 1585–1588 (2006)
21. Wickham, S.F.J., Bath, J., Katsuda, Y., Endo, M., Hidaka, K., Sugiyama, H., Turberfield, A.J.: A DNA-based molecular motor that can navigate a network of tracks. Nature nanotechnology 7(3), 169–73 (Mar 2012)
22. Wickham, S.F.J., Endo, M., Katsuda, Y., Hidaka, K., Bath, J., Sugiyama, H., Turberfield, A.J.: Direct observation of stepwise movement of a synthetic molecular transporter. Nature nanotechnology 6(3), 166–169 (2011)