# Propagation Models for Computing Biochemical Reaction Networks*

Thomas A. Henzinger
IST Austria, Austria
tah@ist.ac.at

Maria Mateescu
EPFL, Switzerland
maria.mateescu@epfl.ch

## ABSTRACT

We introduce *propagation models*, a formalism designed to support general and efficient data structures for the transient analysis of biochemical reaction networks. We give two use cases for propagation abstract data types: the uniformization method and numerical integration. We also sketch an implementation of a propagation abstract data type, which uses abstraction to approximate states.

## 1. INTRODUCTION

We address the *transient analysis* for Markov chains that represent biochemical reaction networks, as formulated by Gillespie [3]. This problem reduces to solving the chemical master equation, which is a differential equation that defines how a probability distribution vector over the states of the network evolves with time.

A computational problem usually has more than one algorithm that solves it, and each algorithm may have different implementations. The implementation of an algorithm strongly relies on the data structures that it uses. Here we introduce *propagation models* (PM), with which we associate a *propagation abstract data type* (ADT), a data structure specifically designed to support the implementation of algorithms for the transient analysis of biochemical reaction networks.

A data structure consists of a state space together with a list of operators and their specifications. At any moment a data structure is in a given state, and by applying one of the operators, the state is updated according to the specification of the operator.

In particular, a propagation ADT is defined over a set of nodes, and its state is a *mass vector* over this discrete space. In other words, the state of a propagation ADT assigns a mass value to each node. The data structure comes with two operations, the **init** operator and the **next** operator. The first one is used at initialization; the second, to let time advance by moving mass from all nodes to their respective neighbors.

We analyze our data structure from two points of view: first, we look at its usefulness, and second, at its efficiency. We present two standard numerical algorithms for the transient analysis of Markov chains which can be implemented using a propagation ADT: the uniformization method and numerical integration. We then sketch a possible implementation for the states and operators of a propagation ADT, which uses abstraction functions to deal with large node sets, such as threshold abstraction for mass values.

## 2. CHEMICAL MASTER EQUATION

A continuous-time Markov chain (CTMC) is a stochastic process $X(t)_{t\geq 0}$ on a discrete set $\mathcal{S}$ of nodes. The stochastic process $X$ is determined by a function $\alpha\colon\ \mathcal{S}\times\mathcal{S}\to\mathbb{R}_{\geq 0}$ which associates a propensity with each pair of nodes, such that, for an infinitesimal time interval $dt$,

$$Pr(X(t+dt)=x' \mid X(t)=x) \ = \ \alpha_{x\to x'}\cdot dt.$$

The transient solution of the Markov chain consists of a probability distribution vector $p^{(t)}$ on $\mathcal{S}$, with

$$p_x^{(t)} = Pr(X(t)=x),$$

which can be computed as the solution of the differential equation:

$$\frac{dp_{x'}^{(t)}}{dt} \ = \ \sum_{x\in\mathcal{S}} \alpha_{x\to x'}\cdot p_x^{(t)} - \sum_{x''\in\mathcal{S}} \alpha_{x'\to x''}\cdot p_{x'}^{(t)}. \qquad (1)$$

In the setting of biochemical reaction networks this is known as the *chemical master equation* (CME).

The solution of Equation 1 is $p^{(t)} = p^{(0)}\cdot e^{Q\cdot t}$, where $Q$ is the generator matrix of the Markov chain, defined by $Q_{x\to x'} = \alpha_{x\to x'}$ if $x\neq x'$, and $Q_{x\to x} = -\sum_{x'\neq x}\alpha_{x\to x'}$.

## 3. DEFINITION

We give formal definitions for both propagation models and propagation ADTs.

Definition 1 (Propagation Model). *A propagation model is a tuple $\langle\mathcal{S},\mathcal{M},\zeta,\pi\rangle$ where*

- *$\mathcal{S}$ is a discrete set of* nodes;[1]

[1]Each node of a propagation model corresponds to a state

- $\mathcal{M}$ *is a set of* mass values*;*
- $\zeta \in [\mathcal{S} \to \mathcal{M}]$ *is the* initialization vector*, which assigns an initial mass value to each node;*
- $\pi$*:* $\mathcal{S} \times \mathcal{S} \times \mathcal{M} \to \mathcal{M}$ *is the* mass propagation function*, which assigns a mass value to each pair of nodes given a mass value for the first node in the pair. For the propagation value from node $x$ to node $x'$, given mass value $\mu$ of $x$, we use the notation $\pi_{x \to x'}(\mu)$.*

EXAMPLE 1 (PREDATOR-PREY). *Consider a predator prey example, where two species, predator $R$ and prey $Y$, interact according to the following reactions:*

$$\emptyset \to Y;\ Y \to R;\ R \to \emptyset$$

*The propagation model that represents the stochastic behavior of the predator-prey example with initial state $\mathbf{y}$ is defined over the set $\mathcal{S} = \mathbb{N}^2_{\geq 0}$ of nodes, where each node $\mathbf{x} = (x_R, x_Y)$ is defined by the number $x_R$ of predators and the number $x_Y$ of prey. The mass values $\mathcal{M} = [0, 1]$ represent probabilities. Furthermore, if $c_1$, $c_2$, $c_3$ are the rate constants of our three reactions:*

- $\zeta_{\mathbf{y}} = 1$ *for the initial state $\mathbf{y}$, and $\zeta_{\mathbf{x}} = 0$ otherwise;*
- $\pi_{(x_R, x_Y) \to (x_R, x_Y + 1)}(p) = p \cdot c_1 \cdot x_Y$*;*
- $\pi_{(x_R, x_Y) \to (x_R + 1, x_Y - 1)}(p) = p \cdot c_2 \cdot x_R \cdot x_Y$*;*
- $\pi_{(x_R, x_Y) \to (x_R - 1, x_Y)}(p) = p \cdot c_3 \cdot x_R$*.*

While the mass values of propagation models usually represent node probabilities (as in the example above), this is not always the case; for example, mass may also represent the expectation of a node variable.

We associate with each propagation model an abstract data type, called a propagation ADT.

DEFINITION 2 (PROPAGATION ABSTRACT DATA TYPE). *The propagation ADT associated with a given propagation model $\langle \mathcal{S}, \mathcal{M}, \zeta, \pi \rangle$ is a triple $\langle \mathbf{state}, \mathbf{init}, \mathbf{next} \rangle$, where*

- *the* state space $\mathbf{state} = [\mathcal{S} \to \mathcal{M}]$ *is the set of vectors that assign to each node in $\mathcal{S}$ a mass value in $\mathcal{M}$;*
- *the* initialization operator $\mathbf{init} \in \mathbf{state}$ *returns $\zeta$, the initialization vector of the propagation model;*
- *the* successor operator $\mathbf{next}$*:* $\mathbf{state} \to \mathbf{state}$ *updates a state in* $\mathbf{state}$ *by propagating mass from every node in $\mathcal{S}$ to its neighbors. Formally, for each state $\mathbf{s} \in \mathbf{state}$ and each node $x' \in \mathcal{S}$,*

$$\mathbf{next}(\mathbf{s})_{x'} = \sum_{x \in \mathcal{S}} \pi_{x \to x'}(\mathbf{s}_x) - \sum_{x \in \mathcal{S}} \pi_{x' \to x''}(\mathbf{s}_{x'}).$$

Note that a propagation ADT supports the use of value iteration algorithms [1].

of a Markov chain. We use the term "node" in order to distinguish it from the state of a propagation ADT, which is a mass vector.

**Algorithm 1 Uniformization**

**Input:** propagation model $N^u$, error $\epsilon$, time horizon $T$;
**Output:** mass vector $p^{(T)}$;
**Variables:** $\mathbf{s}$ of type $\mathbf{prop\_t}^u$;
1: $\mathbf{s} \leftarrow \mathbf{init}^u$;
2: $sum \leftarrow 0$;
3: **while** $sum < 1 - \epsilon$ **do**
4: $\quad p^{(T)} \leftarrow p^{(T)} + \mathcal{P}_{\Lambda t}(k) \cdot \mathbf{s}$;
5: $\quad sum \leftarrow sum + \mathcal{P}_{\Lambda t}(k)$;
6: $\quad \mathbf{s} \leftarrow \mathbf{next}^u(\mathbf{s})$;
7: **end while**.

**Algorithm 2 Integration**

**Input:** propagation model $N^i$, time horizon $T$;
**Output:** mass vector $p^{(T)}$;
**Variables:** $\mathbf{s}$ of type $\mathbf{prop\_t}^i$;
1: $\mathbf{s} \leftarrow \mathbf{init}^i$;
2: $t \leftarrow 0$;
3: **while** $t < T$ **do**
4: $\quad \mathbf{s} \leftarrow \mathbf{next}^i(\mathbf{s})$;
5: $\quad t \leftarrow t + dt$;
6: **end while**;
7: $p^{(T)} \leftarrow \mathbf{s}$.

# 4. APPLICATIONS

We present two algorithms that can be implemented using propagation ADTs, the uniformization method [5] and Runge-Kutta integration, both of which solve the CME (see Equation 1). For the Runge-Kutta method we restrict our presentation to first-order Runge-Kutta in order to keep the presentation simple; propagation models can be used for higher-order methods as well [6].

The uniformization method for Markov chains computes the transient solution $p^{(t)}$ as $\sum_{k=0}^{\infty} r^{(k)} \cdot \mathcal{P}_{\Lambda t}(k)$, which is then approximated by the finite sum $\sum_{k=0}^{R} r^{(k)} \cdot \mathcal{P}_{\Lambda t}(k)$, for a sufficiently large truncation point $R$. The vector $r^{(k)}$ is recursively defined by $r^{(0)} = p^{(0)}$, and $r^{(k+1)} = r^{(k)} \cdot P$, where $P = I + \frac{1}{\Lambda} \cdot Q$ for a uniformization rate $\Lambda$ with $\Lambda \geq max\{-Q_{s,s}\}$. Furthermore, $\mathcal{P}_{\Lambda t}(k)$ is the value of the Poisson distribution with rate $\Lambda \cdot t$, at point $k$.

Suppose we are given a Markov chain with node set $\mathcal{S}$ and propensity rate function $\alpha$: $\mathcal{S} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$. Also given are an initial node $y \in \mathcal{S}$ and a time horizon $T$. We want to compute $p^{(T)}$, knowing that $p^{(0)}_y = 1$. The uniformization method for solving this problem uses the propagation model $N^u = \langle \mathcal{S}, [0, 1], \zeta, \pi^u \rangle$, with $\zeta_s = 1$ if $s = y$ and $\zeta_s = 0$ otherwise, and $\pi^u_{s \to s'}(\mu) = P_{s \to s'} \cdot \mu$. The corresponding propagation ADT $\mathbf{prop\_t}^u$ associated with $N^u$ is used in Algorithm 2. The truncation point $R$ is chosen such that $R$ is the smallest integer for which $\sum_{k=1}^{R} \mathcal{P}_{\Lambda t}(k) > 1 - \epsilon$, for some given error $\epsilon > 0$.

Next, we present a second algorithm, based on a simple integration of Equation 1, using the Euler method (first-order Runge-Kutta) with time step $dt$. For this we define a second propagation model, $N^i = \langle \mathcal{S}, [0, 1], \zeta, \pi^i \rangle$, with the mass propagation function $\pi^i_{s \to s'}(\mu) = \alpha_{s \to s'} \cdot \mu \cdot dt$.

Here we illustrated only discrete-time propagation models, but the CME itself can be viewed as a continuous-time propagation model [6].

**Algorithm 3 Successor**

**Input:** state $\mathbf{s}$, propagation model $N = \langle \mathcal{S}, \mathcal{M}, \zeta, \pi \rangle$, mass threshold $\delta > 0$;
**Output:** state $\mathbf{s}$, error $\epsilon$;
1: *propagate_and_add*$(\mathbf{s}, N)$;
2: *collect_and_remove*$(\mathbf{s}, \delta)$;
3: $\epsilon \leftarrow 1 - sum(\mathbf{s}.\mu)$;
4: **return** $\langle \mathbf{s}.\mu, \epsilon \rangle$.

# 5. IMPLEMENTATION

We briefly describe a possible implementation of a propagation ADT. Since biochemical reaction networks usually have a very large, even infinite node set, it is often advantageous to implement the states of a propagation ADT approximately.

We can obtain approximations of the state of a propagation ADT by applying a threshold abstraction for mass values [2], which considers only nodes that hold significant mass, and/or node aggregation [4], which aggregates many nodes into a single "abstract node" and propagates, in addition to probabilities, also conditional expectations of the node variables. Here we present only the threshold abstraction method.

Algorithm 3 gives the pseudocode for an implementation of the **next** operator, which relies on threshold abstraction. The structure $\mathbf{s}$ holds a state of the propagation model, which is received as input, updated, and returned as output. Each node $n \in \mathbf{s}$ has three different fields: the field $n.x$ refers to a node in $\mathcal{S}$ (the discrete space of the propagation model), the field $n.\mu$ holds the mass value currently associated with the node, and the field $n.acc$ is used as a buffer to temporarily store incoming mass during the successor computation.

The successor computation proceeds in two phases, a propagate and a collect phase. The function *propagate_and_add* performs the propagate phase, during which mass is moved from each node to its successors. In order to do so, for all nodes $n \in \mathcal{S}$, if $\alpha_{n.x \to x'} > 0$, we lookup in $\mathbf{s}$ for the node $n'$ with $n'.x = x'$, and if no such node is found then it is created and added to the current state $\mathbf{s}$. Once the node $n'$ is obtained, we transfer mass from $n$ to $n'$:

$$n'.acc \leftarrow n'.acc + \pi_{n.x \to x'}(n.\mu);$$

$$n.acc \leftarrow n.acc - \pi_{n.x \to x'}(n.\mu).$$

Note that the nodes in the structure $\mathbf{s}$ are created "on-the-fly": if a successor $n'$ does not yet exist in $\mathbf{s}$, it is added dynamically when it obtains a positive mass value. The function *collect_and_remove* performs the collect phase, during which mass is moved from the temporary buffer *acc*, where it has accumulated:

$$n.\mu \leftarrow n.\mu + n.acc.$$

In addition, all nodes that have an insignificant mass value, relative to a given threshold $\delta > 0$, are deleted from the structure $\mathbf{s}$.

The algorithm also returns the accumulated error, computed as $\epsilon = 1 - \sum_{n \in \mathbf{s}} n.\mu$.

# 6. REFERENCES


[1] K. Chatterjee and T. A. Henzinger. Value iteration. In O. Grumberg and H. Veith, editors, *25 Years of Model Checking*, volume 5000 of *Lecture Notes in Computer Science*, pages 107–138. Springer, 2008.

[2] F. Didier, T. A. Henzinger, M. Mateescu, and V. Wolf. Fast adaptive uniformization of the chemical master equation. In *Proceedings of the 2009 International Workshop on High Performance Computational Systems Biology*, HIBI '09, pages 118–127, Washington, DC, USA, 2009. IEEE Computer Society.

[3] D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188:404–425, 1992.

[4] T. A. Henzinger and M. Mateescu. Tail approximation the chemical master equation. *8th International Workshop on Computational Systems Biology*, 2011.

[5] A. Jensen. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuarietidskrift*, 36:87–91, 1953.

[6] M. Mateescu. *Propagation Models for Biochemical Reaction Networks*. Phd thesis, EPFL, Switzerland, 2011.