Termination Problems in Chemical Kinetics

Gianluigi Zavattaro[1] and Luca Cardelli[2]

[1] Dip. Scienze dell'Informazione, Università di Bologna, Italy
[2] Microsoft Research, Cambridge, UK

Abstract. We consider nondeterministic and probabilistic termination problems in a process algebra that is equivalent to basic chemistry. We show that the existence of a terminating computation is decidable, but that termination with any probability strictly greater than zero is undecidable. Moreover, we show that the fairness intrinsic in stochastic computations implies that termination of all computation paths is undecidable, while it is decidable in a nondeterministic framework.

1 Introduction

We investigate the question of whether basic chemical kinetics (kinetics of unary and binary chemical reactions), formulated as a process algebra, is capable of general computation. In particular, we investigate nondeterministic and probabilistic termination problems in the Chemical Ground Form (CGF): a process algebra recently proposed for the compositional description of chemical systems, and proved to be both stochastically and continuously equivalent to chemical kinetics (see [2] for the formal proof of equivalence between CGF and chemical kinetics). The answers to those termination problems reveal a surprisingly rich picture of what is decidable and undecidable in basic chemistry.

We consider three variants of the termination problem: *existential*, *universal*, and *probabilistic* termination. By existential termination we mean the existence of a terminating computation, by universal termination we mean that all possible computations terminate (in a probabilistic setting, by possible computation we mean that the computation has probability > 0), by probabilistic termination we mean that with probability strictly greater than a given ϵ, with $0 < \epsilon < 1$, a terminating computation is executed. We prove that, in the stochastic semantics of CGF, existential termination is decidable, while both probabilistic and universal termination are undecidable. In contrast, in a nondeterministic interpretation of the CGF that abstracts from reaction rates, both existential and universal termination are decidable. This means that: (a) chemical kinetics is not Turing complete, (b) chemical kinetics is Turing complete up to any degree of precision, (c) existential termination is equally hard (decidable) in stochastic and nondeterministic systems, (d) universal termination is harder (undecidable) in stochastic systems than in nondeterministic systems, (e) the fairness implicit in stochastic computations makes checking universal termination undecidable.

In recent work, Soloveichik et al. [8], prove the non-Turing completeness of Stochastic Chemical Reaction Networks (which are equivalent to the CGF [2])

by reduction to the decidability of chemical state coverability, which they call reachability. We prove more strongly that exact chemical state reachability is also decidable, as well as that existential termination and boundedness are decidable. (All these argument are based on decidability results in Petri Nets.) The same authors also prove the possibility of approximating RAM and Turing Machine computations up to an arbitrarily small error ϵ. Their encodings allow them to prove the undecidability of probabilistic coverability. We prove the undecidability of probabilistic termination, probabilistic reachability, probabilistic boundedness, and of universal termination. There are technical differences in our RAM encodings that guarantee the stronger results. For example, terminating computations are still terminating in our encoding of RAMs, while in [8] a "clock" process keeps running even after termination of the main computation.

2 Chemical Ground Form

In the CGF each species has an associated definition describing the possible actions for the molecules of that species. Each action $\pi_{(r)}$ has an associated stochastic rate r (a positive real number) which quantifies the expected execution time for the action π. Action $\tau_{(r)}$ indicates the possibility for a molecule to be engaged in a unary reaction. For instance, the definition $A = \tau_{(r)}; (B|C)$ says that one molecule of species A can be engaged in a unary reaction that produces two molecules, one of species B and one of species C (the operator "|" is borrowed from process algebras such as CCS [6], where it represents parallel composition, and corresponds here to the chemical "+"). Binary reactions have two reactants. The two reactants perform two complementary actions $?a_{(r)}$ and $!a_{(r)}$, where a is a name used to identify the reaction; both the name a and the rate r must match for the reaction to be enabled. For instance, given the definitions $A = ?a_{(r)}; C$ and $B = !a_{(r)}; D$, we have that two molecules of species A and B can be engaged in a binary reaction that produces two molecules, one of species C and one of species D. If the molecules of one species can be engaged in several reactions, then the corresponding definition admits a choice among several actions. The syntax of choice is as follows: $A = \tau_{(r)}; B \oplus ?a_{(r')}; C$, meaning that molecules of species A can be engaged in either a unary reaction, or in a binary reaction with another molecule able to execute the complementary action $!a_{(r')}$.

Definition 1 (Chemical Ground Form (CGF)). *Consider the following denumerable sets:* Species *ranged over by variables* $X, Y, \cdots$, Channels *ranged over by* $a, b, \cdots$, *Moreover, let* $r, s, \cdots$ *be rates (i.e. positive real numbers).*
The syntax of CGF is as follows (where the big $\big|$ *separates syntactic alternatives while the small* $|$ *denotes parallel composition):*

$$
\begin{array}{rcll}
E & ::= & \mathbf{0} \;\big|\; X = M, E & \textit{Reagents} \\
M & ::= & \mathbf{0} \;\big|\; \pi; P \oplus M & \textit{Molecule} \\
P & ::= & \mathbf{0} \;\big|\; X|P & \textit{Solution} \\
\pi & ::= & \tau_{(r)} \;\big|\; ?a_{(r)} \;\big|\; !a_{(r)} & \textit{Internal, Input, Output prefix} \\
CGF & ::= & (E, P) & \textit{Reagents and initial Solution}
\end{array}
$$

Given a CGF (E, P), we assume that all variables in P occur also in E. Moreover, for every variable X in E, there is exactly one definition $X = M$ in E.

In the following, trailing $\mathbf{0}$ are left implicit, and we use $|$ also as an operator over the syntax: if P and P' are $\mathbf{0}$-terminated lists of variables, according to the syntax above, then $P|P'$ means appending the two lists into a single $\mathbf{0}$-terminated list. Thus, if P is a solution, then $\mathbf{0}|P$, $P|\mathbf{0}$, and P are syntactically equal. The solution composed of k instances of X is denoted with $\prod_k X$.

We consider the discrete state semantics for the CGF defined in [2] in terms of Continuous Time Markov Chains (CTMCs). The states of the CTMCs are solutions in normal form denoted with $P^\dagger$: for a solution P, we indicate with $P^\dagger$ the normalized form of P where the variables are sorted in lexicographical order (with $\mathbf{0}$ at the end), possibly with repetitions. The CTMC associated to a chemical ground form is obtained in two steps: we first define the Labeled Transition Graph (LTG) of a chemical ground form, then we show how to extract a CTMC from the labeled transition graph.

We use the following notation. Let $E.X$ be the molecule defined by X in E, and $M.i$ be the i-th summand in a molecule of the form $M = \pi_1; P_1 \oplus \cdots \oplus \pi_n; P_n$. Given a solution in normal form $P^\dagger$, with $P^\dagger.m$ we denote the m-th variable in $P^\dagger$, with $P^\dagger \setminus (m_1, \cdots, m_n)$ we denote the solution obtained by removing from $P^\dagger$ the m_i-th molecule for each $i \in \{1, \cdots, n\}$.

A *Labeled Transition Graph* (LTG) is a set of quadruples $\langle l : S^\dagger \xrightarrow{r} T^\dagger \rangle$ where the transition labels l are either of the form $\{m.X.i\}$ or $\{m.X.i, n.Y.j\}$, where m, n, i, j are positive integers, X, Y are species names, $m.X.i$ are ordered triples and $\{\cdots, \cdots\}$ are unordered pairs.

Definition 2 (Labeled Transition Graph (LTG) of a Chemical Ground Form). *Given the Chemical Ground Form (E, P), we define $Next(E, P)$ as the set containing the following kinds of labeled transitions:*

- $\langle \{m.X.i\} : P^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $P^\dagger.m = X$ *and* $E.X.i = \tau_{(r)}; Q$ *and* $T = (P^\dagger \setminus m)|Q$;
- $\langle \{m.X.i, n.Y.j\} : P^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $P^\dagger.m = X$ *and* $P^\dagger.n = Y$ *and* $m \neq n$ *and* $E.X.i = ?a_{(r)}; Q$ *and* $E.Y.j = !a_{(r)}; R$ *and* $T = (P^\dagger \setminus m, n)|Q|R$.

The Labeled Transition Graph of (E, P) is defined as follows:

$$LTG(E, P) = \bigcup_n \Psi_n$$
$$\text{where } \Psi_0 = Next(E, P) \text{ and } \Psi_{n+1} = \bigcup \{Next(E, Q) \mid Q \text{ is a state of } \Psi_n\}$$

We now define how to extract from an LTG the corresponding CTMC.

Definition 3 (Continuous Time Markov Chain associated to an LTG). *If Ψ is an LTG, then $|\Psi|$ is the associated CTMC, defined as the set of the triples $P \overset{r}{\mapsto} Q$ with $P \neq Q$, obtained by summing the rates of all the transitions in Ψ that have the same source and target state: $|\Psi| = \{P \overset{r}{\mapsto} Q \text{ s.t. } \exists \langle l : P \xrightarrow{r'} Q \rangle \in \Psi \text{ with } P \neq Q, \text{ and } r = \sum r_i \text{ s.t. } \langle l_i : P \xrightarrow{r_i} Q \rangle \in \Psi\}$.*

It is worth noting that two solutions $Q^\dagger$ and $R^\dagger$ are connected by a transition in $LTG(E,P)$ if and only if they are connected by a transition in $|LTG(E,P)|$. In fact, the transitions of the latter are achieved by collapsing into one transition those transitions of the former that share the same source and target solutions. The rate of the new transition is the sum of the rates of the collapsed transitions.

Given a CGF (E,P), a *computation* is a sequence of transitions in the CTMC $|LTG(E,P)|$ starting with a transition with source solution $P^\dagger$, and such that the target solution of one transition coincides with the source state of the next transition. We say that a solution Q is *reachable* in (E,P) if there exists a computation with $Q^\dagger$ as the target solution of the last transition. A solution Q is *terminated* in $|LTG(E,P)|$ if $Q^\dagger$ has no outgoing transitions.

The CTMC semantics of CGF defines a probabilistic interpretation of the behavior of a CGF (E,P): given any solution $T^\dagger$ of $|LTG(E,P)|$, if it has n outgoing transitions labeled with $r_1, \cdots, r_n$, the probability that the j-th transition is taken is $r_j/(\sum_i r_i)$. Thus, we can associate probability measures (we consider the standard probability measure for Markov chains —see e.g. [5]) to computations in $|LTG(E,P)|$. We use this technique to define the three variants of the termination problem we consider in this paper.

Definition 4 (Existential, universal and probabilistic termination). *Consider a CGF (E,P) and its CTMC $|LTG(E,P)|$. Let p be the probability measure associated to the computations in $|LTG(E,P)|$ leading to a terminated solution. We say that (E,P)* existentially terminates *if $p > 0$, (E,P)* universally terminates *if $p = 1$, (E,P)* probabilistically terminates *with probability higher than ϵ (for $0 < \epsilon < 1$) if $p > \epsilon$.*

We will consider also probabilistic variants of other properties. Consider a CGF (E,P), its CTMC $|LTG(E,P)|$, and a real number ϵ such that $0 \leq \epsilon < 1$. We say that a solution Q is *ϵ-reachable* if the probability measure of the computations in $|LTG(E,P)|$ leading to $Q^\dagger$ is $> \epsilon$. We say that (E,P) is *ϵ-bound* if the set of ϵ-reachable solutions is finite. We say that (E,P) is *ϵ-terminating* if the probability measure of the computations in $|LTG(E,P)|$ leading to a terminated solution is $> \epsilon$. We say that (E,P) is *ϵ-diverging* if the probability measure of the infinite computations in $|LTG(E,P)|$ is $> \epsilon$.

It is worth noting that, in a probabilistic setting, existential termination coincides with 0-termination, universal termination with the negation of 0-divergence, and probabilistic termination with ϵ-termination for $\epsilon > 0$.

3 Decidability Results

In this section we resort to a Place/Transition Petri net (P/T net) semantics for CGF, that can be interpreted as a purely nondeterministic semantics of CGF that abstracts away from the stochastic rates. In this purely nondeterministic framework several properties are decidable. In fact, in P/T nets, properties such as *reachability* (the existence of a computation leading to a given state), *boundedness* (the finiteness of the set of reachable states), *termination* (reachability of

a deadlocked state), and *divergence* (the existence of an infinite computation) are decidable (see [4] for a survey on decidable properties for Petri Nets).

Definition 5 (Place/Transition Net). *A P/T net is a tuple $N = (S, T)$ where S is the set of* places*, $\mathcal{M}_{fin}(S)$ is the set of the finite multisets over S (each of which is called a* marking*) and $T \subseteq \mathcal{M}_{fin}(S) \times \mathcal{M}_{fin}(S)$ is the set of* transitions*. A transition (c, p) is written $c \Rightarrow p$. The marking c, represents the tokens to be "consumed"; the marking p represents the tokens to be "produced". A transition $c \Rightarrow p$ is* enabled *at a marking m if $c \subseteq m$. The execution of the transition produces the marking $m' = (m \setminus c) \oplus p$ (where $\setminus$ and $\oplus$ are the difference and the union operators on multisets). This is written as $m[\rangle m'$. A* dead marking *is a marking in which no transition is enabled. A* marked *P/T net is a tuple $N(m_0) = (S, T, m_0)$, where (S, T) is a P/T net and m_0 is the* initial marking*. A computation in $N(m_0)$ leading to the marking m is a sequence $m_0[\rangle m_1[\rangle m_2 \cdots m_n[\rangle m$.*

Given a CGF (E, P), we define a corresponding P/T net $N = (S, T)$ and a corresponding marked P/T net $N(m_0)$. We first need to introduce an auxiliary function $Mark(P)$ that associates to a solution P the multiset of its variables:

$$Mark(P) \;=\; \begin{cases} \emptyset & \text{if } P = \mathbf{0} \\ \{X\} \oplus Mark(P') & \text{if } P = X|P' \end{cases}$$

Definition 6 (Net of a CGF). *Given a CGF (E, P), with $Net_{(E,P)}$ we denote the corresponding P/T net (S, T) where:*

$$\begin{aligned} S &= \{X \mid X \text{ occurs in } E\} \\ T &= \big\{\ \{X\} \Rightarrow Mark(X_1|\cdots|X_n) \mid \\ &\qquad E.X.i = \tau_{(r)}; (X_1|\cdots|X_n)\ \big\} \cup \\ &\quad \big\{\ \{X, Y\} \Rightarrow Mark(X_1|\cdots|X_n) \oplus Mark(Y_1|\cdots|Y_m) \mid \\ &\qquad E.X.i = ?a_{(r)}; (X_1|\cdots|X_n) \text{ and } E.Y.j = !a_{(r)}; (Y_1|\cdots|Y_m)\ \big\} \end{aligned}$$

The corresponding marked P/T net is $Net_{(E,P)}(Mark(P))$.

Note that the set of places S corresponds to the set of variables X defined in E, the transitions represents the possible actions, and the initial marking is the multiset of variables in the solution P. It is also worth observing that in the net semantics we do not consider the rates (r) of the actions.

We now formalize the correspondence between the behaviors of a CGF and of its corresponding P/T net.

Theorem 1. *Consider a CGF (E, P) and the corresponding P/T net $Net_{(E,P)} = (S, T)$. We have that:*

1. *if $\langle l : P^\dagger \xrightarrow{r} Q^\dagger \rangle$ is in $Next(E, P)$ (for some l and r) then we have also that $Mark(P)[\rangle Mark(Q)$ in $Net_{(E,P)}$;*
2. *if there exists m such that $Mark(P)[\rangle m$ in $Net_{(E,P)}$, then there exist l, r and Q such that $\langle l : P^\dagger \xrightarrow{r} Q^\dagger \rangle$ is in $Next(E, P)$ and $Mark(Q) = m$.*

Proof (sketch). The proofs of the two statements are by case analysis on the possible transitions in $Next(E, P)$ as defined in the Definition 2 —for the first statement— or on the possible transitions enabled in $Mark(P)$ as defined in the Definition 6 —for the second statement. □

This theorem allows us to conclude that the P/T net semantics faithfully reproduces the standard CGF transitions. The only difference is that it abstracts away from the stochastic rates. For this reason, we consider the P/T net semantics as a purely nondeterministic interpretation of CGF. Reachability, boundedness, termination, and divergence are decidable for P/T nets; thus we can conclude that all these properties are decidable also in the CGF under a purely nondeterministic interpretation.

As a consequence of Theorem 1, existential termination is decidable.

Theorem 2. *Consider a CGF (E, P). We have that (E, P) existentially terminates if and only if a dead marking is reachable in the $Net_{(E,P)}(Mark(P))$.*

Proof. It is easy to see from the definition of $LTG(E, P)$ and $|LTG(E, P)|$ that the latter contains all and only those solutions (in normal form) reachable in (E, P) with a finite number of transitions, each one having a probability > 0 to be chosen. Thus a solution is reachable with probability > 0 if and only if it is in $|LTG(E, P)|$. As a consequence of Theorem 1 we have that a solution $Q^{\dagger}$ is in $|LTG(E, P)|$ if and only if $Mark(Q)$ is reachable in $Net_{(E,P)}(Mark(P))$. Moreover, Theorem 1 also guarantees that Q is terminated if and only if $Mark(Q)$ is a dead marking in $Net_{(E,P)}$ (this proves the theorem). □

As a corollary of Theorem 1 we have that also the probabilistic variants of reachability and boundedness can be reduced to the corresponding properties in the nondeterministic setting. On the contrary, this does not hold for divergence. (This will be discussed in the next section.) We can summarize the results of this section simply saying that ϵ-*termination*, ϵ-*reachability*, and ϵ-*boundedness* are decidable when $\epsilon = 0$.

4 Undecidability Results

This section is divided in two parts. In the first one we prove that probabilistic termination (i.e. ϵ-termination with $\epsilon > 0$) is undecidable. (We also comment on how to show that also ϵ-divergence, ϵ-boundedness, and ϵ-reachability are undecidable when $\epsilon > 0$.) In the second part we prove the undecidability of universal termination (thus also of 0-divergence).

4.1 Undecidability of Probabilistic Termination

We prove the undecidability of probabilistic termination showing how to approximately model in CGF the behavior of any Random Access Machines (RAMs) [7], a well known register based Turing powerful formalism. More precisely, we reduce the termination problem for RAMs to the probabilistic termination with probability higher than any ϵ such that $0 < \epsilon < 1$.

We first recall the definition of Random Access Machines.

Definition 7 (Random Access Machines (RAMs)). *A RAM $\mathcal{R}$ is composed of a set of registers $r_1, \cdots, r_m$ that contain non negative integer numbers and a set of indexed instructions $I_1, \cdots, I_n$ of two possible kinds:*

- $I_i = Inc(r_j)$ *that increments the register r_j and then moves to the execution of the instruction with index $i+1$ and*
- $I_i = DecJump(r_j, s)$ *that attempts to decrement the register r_j; if the register does not hold 0 then the register is actually decremented and the next instruction is the one with index $i+1$, otherwise the next instruction is the one with index s.*

We use the following notation: $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ represents the state of the computation of the RAM which is going to execute the instruction I_i with registers that contain $l_1, \cdots, l_m$, respectively; $(I_i, r_1 = l_1, \cdots, r_m = l_m) \mapsto (I_j, r_1 = l'_1, \cdots, r_m = l'_m)$ describes one step of computation of the RAM; $(I_i, r_1 = l_1, \cdots, r_m = l_m)\downarrow$ denotes final states of the computation in which I_i is undefined. Without loss of generality, we assume the existence of a special index *halt* such that all final states contain an instruction with that index, namely $(I_i, r_1 = l_1, \cdots, r_m = l_m)\downarrow$ if and only if $i = halt$.

The basic idea that we follow in modeling RAMs in CGF is to use one species I^i for each instruction I_i, and one species R^j for each register r_j. The state $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ of the RAM is modeled by a solution that contains one molecule of species I^i, l_1 molecules of species R^1, $\cdots$, and l_m molecules of species R^m (plus a certain amount of inhibitor molecules of species Inh, whose function will be discussed below). The behavior of the molecules of species I^i is to update the register according to the corresponding instruction I_i, and to activate the execution of the next instruction I_j by producing the molecule of species I^j.

An $Inc(r_j)$ instruction simply produces one molecule of species R^j. On the other hand, a $DecJump(r_j, s)$ instruction should test the absence of molecules of species R^j before deciding whether to execute the jump, or to consume one of the available molecules of that species. As it is not possible to verify the absence of molecules, we admit the execution of the jump even if molecules of species R^j are available. In this case, we say that a *wrong jump* is executed. In order to reduce the probability of wrong jumps, we put their execution in competition with alternative behaviors involving the inhibitor molecules in such a way that the greater is the quantity of inhibitor molecules in the solution, the smaller is the probability to execute a wrong jump.

We are now ready to formally define our encoding of RAMs.

Definition 8. *Given a RAM $\mathcal{R}$ and one of its states $(I_i, r_1 = l_1, \cdots, r_m = l_m)$, let $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h$ denote the solution:*

$$I^i \mid \prod_{l_1} R^1 \mid \cdots \mid \prod_{l_m} R^m \mid \prod_{h} Inh$$

where:

$$I^i \;=\; \begin{cases} \tau;(I^{i+1}|R_j) & \text{if } I_i = Inc(r_j) \\ !r_j;(I^{i+1}|Inh) \;\oplus\; \tau;C^2_{i,s} & \text{if } I_i = DecJump(r_j,s) \\ \mathbf{0} & \text{if } I_i = I_{halt} \end{cases}$$

$$C^2_{i,s} \;=\; !inh;I^i \;\oplus\; \tau;C^1_{i,s} \qquad C^1_{i,s} \;=\; !inh;I^i \;\oplus\; \tau;I^s$$

$$R^j \;=\; ?r_j;\mathbf{0} \qquad Inh \;=\; ?inh;Inh$$

Note that h is used to denote the number of occurrences of the molecules of species Inh. We take all subscripts action rates equal to 1 and we omit them (this choice allows us to simplify the proof of Proposition 1). In the following, we use $E_{\mathcal{R}}$ for the set of the above definitions of species I^i, $C^2_{i,s}$, $C^1_{i,s}$, R^j, Inh.

Note that before actually executing a jump, two internal τ actions must be executed in sequence (those in the definition of the species $C^2_{i,s}$ and $C^1_{i,s}$), and both of them are in competition with the action $!inh$ willing to perform an interaction with one of the inhibitor molecules of species Inh. Thus, the higher is the number of inhibitor molecules, the smaller is the probability to perform this sequence of two internal actions.

We now formalize the correspondence between the behavior of a RAM and of its encoding in CGF.

Proposition 1. *Let $\mathcal{R}$ be a RAM. Given one of its states $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ and $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h$, for any h, we have:*

1. *if $I_i = I_{halt}$ then $(I_i, r_1 = l_1, \cdots, r_m = l_m)\downarrow$ and $Next(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ has no transitions;*
2. *if $I_i = Incr_j$ or $I_i = DecJump(r_j, s)$ with $l_j = 0$ and $(I_i, r_1 = l_1, \cdots, r_m = l_m) \mapsto (I_j, r_1 = l'_1, \cdots, r_m = l'_m)$, then the solution $[\![(I_j, r_1 = l'_1, \cdots, r_m = l'_m)]\!]^\dagger_h$ is reachable in $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ with probability $= 1$;*
3. *if $I_i = DecJump(r_j, s)$ with $l_j > 0$ and $(I_i, r_1 = l_1, \cdots, r_m = l_m) \mapsto (I_j, r_1 = l'_1, \cdots, r_m = l'_m)$, then the solution $[\![(I_j, r_1 = l'_1, \cdots, r_m = l'_m)]\!]^\dagger_{h+1}$ is reachable in $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ with probability $> 1 - \frac{1}{h^2}$.*

Proof. If $I_i = I_{halt}$ or $I_i = Inc(r_j)$ the corresponding statements (the first one and the first part of the second one) are easy to prove. We detail the proof only for $I_i = DecJump(r_j, s)$.

If r_j is empty, the probability measure for the computations in $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ passing through $[\![(I_j, r_1 = l'_1, \cdots, r_m = l'_m)]\!]^\dagger_h$ is (see Figure 1):

$$\sum_{i=0}^{\infty} \big(\frac{h}{h+1} + \frac{1}{h+1} \times \frac{h}{h+1}\big)^i \times \frac{1}{(h+1)^2} = 1$$

If r_j is not empty, i.e. $l_j > 0$, the standard probability measure for the computations passing through $[\![(I_j, r_1 = l'_1, \cdots, r_m = l'_m)]\!]^\dagger_{h+1}$ is (see Figure 1):

$$\sum_{i=0}^{\infty} \big(\frac{1}{l_j+1} \times \frac{h}{h+1} + \frac{1}{l_j+1} \times \frac{1}{h+1} \times \frac{h}{h+1}\big)^i \times \frac{l_j}{l_j+1} > 1 - \frac{1}{h^2}$$

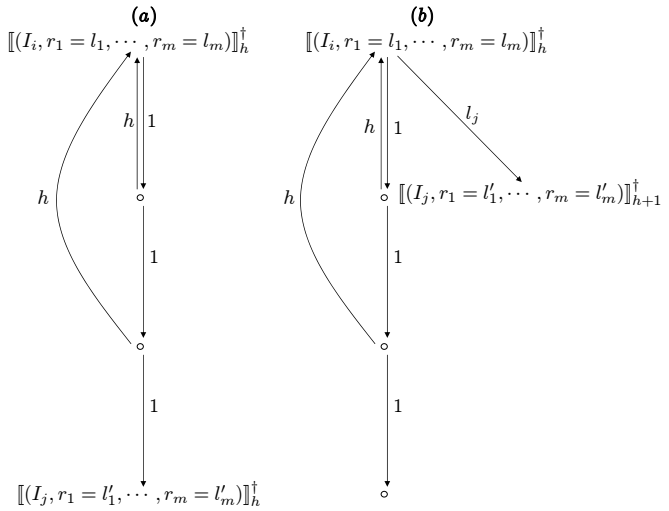

Fig. 1. Fragment of the CTMC $|LTG(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)|$ in case $I_i = DecJump(r_j, s)$ with $l_j = 0$ (a) or $l_j > 0$ (b).

□

The above proposition states the correspondence between a single RAM step and the corresponding encoding in CGF. We conclude that a RAM terminates its computation if and only if a terminated solution is reachable with a probability that depends on the initial number of inhibitor molecules in the encoding.

Theorem 3. *Let $\mathcal{R}$ be a RAM. We have that the computation of $\mathcal{R}$ starting from the state $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ terminates if and only if the CGF $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ probabilistically terminates with probability higher than $1 - \sum_{k=h}^{\infty} \frac{1}{k^2}$.*

Proof. In the light of Proposition 1 we have that only decrement operations are not reproduced with probability $= 1$, but with probability $> 1 - \frac{1}{h^2}$. Moreover, after the execution of a decrement operation, the value h of inhibitor molecules is incremented by one. Thus, a RAM computation including d decrement operations is faithfully reproduced with probability strictly greater than $\prod_{k=h}^{h+d} \left(1 - \frac{1}{k^2}\right) > 1 - \sum_{k=h}^{h+d} \frac{1}{k^2}$. Henceforth, any terminating computation is reproduced with probability strictly greater than $1 - \sum_{k=h}^{\infty} \frac{1}{k^2}$. □

It is well known that the series $\sum_{h=1}^{\infty} \frac{1}{h^2}$ is convergent (to $\frac{\pi^2}{6}$), thus for every small value $\delta > 0$ there exists a corresponding initial amount h of inhibitor molecules such that $\sum_{k=h}^{\infty} \frac{1}{k^2} < \delta$. Henceforth, in order to reduce RAM termination to probabilistic termination with probability higher than any $0 < \epsilon < 1$, it is sufficient to consider an initial value h such that $\sum_{k=h}^{\infty} \frac{1}{k^2} < (1 - \epsilon)$.

The RAM encoding presented in Definition 8 reproduces also unbounded RAM computations with any degree of precision. Thus also ϵ-divergence is undecidable when $\epsilon > 0$. On the contrary, such encoding does not allow us to prove the undecidability of ϵ-boundedness and ϵ-reachability.

We first show how to reduce the RAM divergence problem to ϵ-boundedness. This does not hold for the encoding in the Definition 8 because there exists divergent RAMs with a bounded corresponding CGF. Consider, for instance, the RAM composed of only the instruction $I_1 = DecJump(r_1, 1)$ that performs an infinite loop if the register r_1 is initially empty. It is easy to see that the corresponding CGF is bounded.

In order to guarantee that an infinite RAM computation generates an unbounded CGF, we can simply add a new molecule of a new species A every time a jump is performed. As an infinite RAM computation executes infinitely many jump operations, an unbounded amount of molecules of species A will be generated. The new encoding is defined as in the Definition 8 replacing the definition of the species $C^1_{i,s}$ with the following one: $C^1_{i,s} = !inh; I^i \oplus \tau; (I^s|A)$.

We conclude this section observing how to reduce RAM termination to ϵ-reachability. This does not hold for the above encodings because the solution representing the final state of a RAM computation is not known beforehand. In fact, besides the fact that the final contents of the registers is not known, we have that the final solution will contain a number of inhibitor molecules that depends on the number of decrement operations executed during the computation (as each decrement adds one molecule of species Inh). In order to know beforehand the final solution, we allow the molecule I^{halt} to remove the register molecules of species R^j as well as all the inhibitor molecules of species Inh. In this way, if the computation terminates, we have that the final solution surely contains only the molecule I^{halt}.

Namely, we modify in the Definition 8 the definitions of the species I^{halt} and Inh as follows:

$$\begin{aligned} I^{halt} &= \bigoplus_{j=1}^{m} !r_j; I^{halt} \oplus !remove; I^{halt} \\ Inh &= ?inh; Inh \oplus ?remove; \mathbf{0} \end{aligned}$$

4.2 Undecidability of Universal Termination

The undecidability of universal termination (thus also of 0-divergence) is proved introducing an intermediary nondeterministic computational model, that we call *finitely faulting RAMs* ($\mathcal{FF}$RAMs). This model corresponds to RAMs in which the execution of $DecJump$ instructions is nondeterministic when the tested register is not empty: an $\mathcal{FF}$RAM can either decrement the register or execute a wrong jump. The peculiarity of $\mathcal{FF}$RAMs is that in an infinite computation only finitely many wrong jumps are executed. We first show that it is possible to define an encoding of $\mathcal{FF}$RAMs in CGF such that the universal termination problem for $\mathcal{FF}$RAMs coincides with the universal termination problem for the corresponding CGF. Then we prove the undecidability of the universal termination problem for $\mathcal{FF}$RAMs showing how to reduce the RAM termination problem to the verification of the existence of an infinite computation in

$\mathcal{FF}$RAMs (which corresponds to the complement of the universal termination problem). We start defining *finitely faulting RAMs*.

Definition 9 (Finitely Faulting RAMs ($\mathcal{FF}$RAMs)). Finitely Faulting RAMs *are defined as traditional RAMs (see Definition 7) with the only difference that given an instruction $I_i = DecJump(r_j, s)$ and a RAM state $(I_i, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)$ with $l_j > 0$, two possible computation steps are permitted: $(I_i, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m) \mapsto (I_{i+1}, r_1 = l_1, \cdots, r_j = l_j - 1, \cdots, r_m = l_m)$ and $(I_i, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m) \mapsto (I_s, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)$. The second computation step is called* wrong jump *because a jump is executed even if the tested register is not empty. The peculiar property of $\mathcal{FF}$RAMs is that in every computation (also infinite ones), finitely many wrong jumps are executed.*

We now show how to define an encoding of $\mathcal{FF}$RAMs in CGF such that infinite computations in the $\mathcal{FF}$RAMs computational model corresponds to infinite computation with probability > 0 in the corresponding CGF.

The $\mathcal{FF}$RAM encoding is defined as in Definition 8 adding a transition to a terminated state which can be selected with probability $\geq \frac{1}{2}$ while executing wrong jumps. In this way, we guarantee that in an infinite computation infinitely many wrong jumps cannot be executed because the new transition to the terminated state cannot be avoided indefinitely.

Definition 10 ($\mathcal{FF}$RAM Modeling). *Given a $\mathcal{FF}$RAM $\mathcal{R}$ and one of its states $(I_i, r_1 = l_1, \cdots, r_m = l_m)$, $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h$ is defined as in Definition 8. Also the species I^i, R^j, Inh, and $C^2_{i,s}$ are defined as in Definition 8, while $C^1_{i,s}$ is defined as follows:*

$$C^1_{i,s} \;=\; !inh; I^i \;\oplus\; \tau; C^0_s \qquad\qquad C^0_s \;=\; !r_j; I_{halt} \;\oplus\; \tau; I_s$$

In the following, we use $E_{\mathcal{R}}$ for the new set of definitions of species I^i, $C^2_{i,s}$, $C^1_{i,s}$, C^0_s, R^j, and Inh.

We now revisit the Proposition 1 adapting it to the new encoding.

Proposition 2. *Let $\mathcal{R}$ be a $\mathcal{FF}RAM$. Given one of its states $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ and $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h$, for any h, we have:*

1. *(as in Proposition 1);*
2. *(as in Proposition 1);*
3. *if $I_i = DecJump(r_j, s)$ with $l_j > 0$ then with probability 1 one of the following states are reachable in $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)]\!]_h)$:*
   - *$[\![(I_{i+1}, r_1 = l_1, \cdots, r_j = l_j - 1, \cdots, r_m = l_m)]\!]^{\dagger}_{h+1}$ (with prob. $> 1 - \frac{1}{h^2}$);*
   - *$[\![(I_{halt}, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)]\!]^{\dagger}_h$;*
   - *$[\![(I_s, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)]\!]^{\dagger}_h$ (with probability $0 < p < \frac{1}{2}$).*

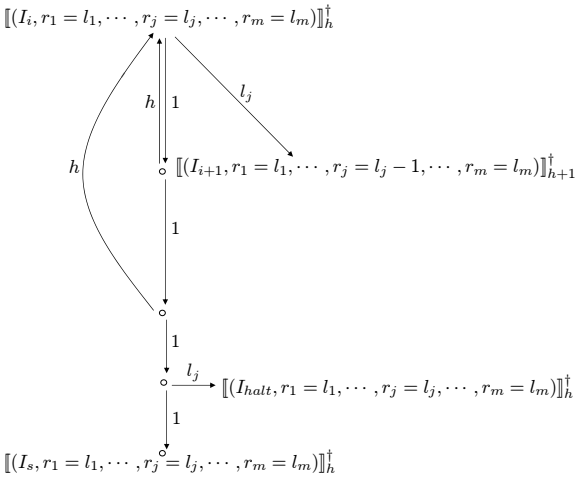

Fig. 2. Fragment of the CTMC $|LTG(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)|$ in case $I_i = DecJump(r_j, s)$ with $l_j > 0$.

Proof. The first two statements are proved as in Proposition 1. We sketch the proof for the third statement. The probability measure for the computations in $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)]\!]_h)$ passing through $[\![(I_{i+1}, r_1 = l_1, \cdots, r_j = l_j - 1, \cdots, r_m = l_m)]\!]^\dagger_{h+1}$ is computed as in Proposition 1.

The probability measure p for the computations passing through $[\![(I_s, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)]\!]^\dagger_h$ is (see Figure 2):

$$\sum_{i=0}^{\infty} \big(\frac{1}{l_j+1} \times \frac{h}{h+1} + \frac{1}{l_j+1} \times \frac{1}{h+1} \times \frac{h}{h+1}\big)^i \times \big(\frac{1}{l_j+1}\big)^2 \times \big(\frac{1}{h+1}\big)^2$$

It is easy to see that as $l_j > 0$, then $p < \frac{1}{2}$. Finally, we observe that the probability measure of the computations leading to $[\![(I_{halt}, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)]\!]^\dagger_h$ is equal to 1 minus the probability measure of the computations passing through either $[\![(I_{i+1}, r_1 = l_1, \cdots, r_j = l_j - 1, \cdots, r_m = l_m)]\!]^\dagger_{h+1}$ or $[\![(I_s, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)]\!]^\dagger_h$. □

The above proposition states the correspondence between a single computation step of a $\mathcal{FF}$RAM and that of the corresponding CGF. We conclude that a $\mathcal{FF}$RAM has an infinite computation if and only if there exists an infinite computation with probability > 0 in the corresponding CGF.

Theorem 4. *Let $\mathcal{R}$ be a $\mathcal{FF}$RAM. We have that $\mathcal{R}$ has an infinite computation starting from the state $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ if and only if the CGF*

$(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ *has an infinite computation for some initial amount* h *of inhibitor molecules.*

Proof. We first consider the *only if* part. Assume the existence of an infinite computation of $\mathcal{R}$ starting from the state $(I_i, r_1 = l_1, \cdots, r_m = l_m)$. This computation will execute infinitely many $DecJump$ instructions, but only finitely many wrong jumps. We now consider the CGF $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ for a generic h. According to the Proposition 2, it can reproduce the same infinite computation with probability strictly greater than $\prod_{k=h}^{\infty}\left(1-\frac{1}{k^2}\right) \times \prod_{k=1}^{w} p_k$ where w is the number of wrong jumps, and p_k is the probability for the k-th wrong jump computed as in Proposition 2. Let p_{min} be the minimum among $p_1, \cdots, p_w$. We have that the above probability is strictly greater than $\left(1-\sum_{k=h}^{\infty}\frac{1}{k^2}\right) \times \left(\frac{1}{p_{min}}\right)^w$. We have already discussed, after Theorem 3, that the series $\sum_{h=1}^{\infty}\frac{1}{h^2}$ is convergent, thus there exists h such that $\sum_{k=h}^{\infty}\frac{1}{k^2} < 1$. If we consider this particular value h, the overall probability for the infinite computation is > 0.

We now consider the *if* part. Assume the existence of an infinite computation with probability > 0 in the CGF $(E_{\mathcal{R}}, [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]_h)$ for some h. This computation corresponds to an infinite computation of $\mathcal{R}$ for the two following reasons. We first observe that the infinite computation reproduces infinitely many correct computation steps $(I_i, r_1 = l_1, \cdots, r_m = l_m) \mapsto (I_j, r_1 = l'_1, \cdots, r_m = l'_m)$ of $\mathcal{R}$. In fact, the unique wrong computation step could be the one described in the second item of the third statement of Proposition 2. This computation step leads to the encoding of the terminated state $(I_{halt}, r_1 = l_1, \cdots, r_j = l_j, \cdots, r_m = l_m)$, but in this case the computation cannot be infinite. Then, we observe that the number of wrong jumps is finite. In fact, if we assume (by contradiction) that the computation contains infinitely many wrong jumps, we have that (in the light of the third item of the third statement of Proposition 2) the probability of the infinite computation is smaller than $\prod_{i=1}^{\infty}\frac{1}{2}$, thus it cannot be > 0. □

We now prove that the existence of an infinite computation in $\mathcal{FF}$RAMs is undecidable by defining an encoding that reduces the termination problem for RAMs to the divergence problem for $\mathcal{FF}$RAMs. As it is not restrictive, we consider only RAMs starting with all registers empty. Our technique has been inspired by a similar one used in [3]. We initially assume that an arbitrary number k of wrong jumps occurs and, as a consequence, the number k is introduced in a special register. Then we let the $\mathcal{FF}$RAM repeat indefinitely the simulation of the behavior of the corresponding RAM, but if this simulation requires more than k steps, the encoding blocks (this is ensured by decrementing the special register before simulating every computational step). In this way, if a RAM terminates, then the corresponding $\mathcal{FF}$RAM (with k greater than the length of the RAM computation) can diverge. On the other hand, if an infinite computation of the $\mathcal{FF}$RAM exists, this has an infinite suffix that does not contain wrong jumps. In this correct part of the computation, the encoding faithfully simulates the RAM computation infinitely often; this is possible only if the RAM terminates.

Theorem 5. *Given a RAM $\mathcal{R}$, there exists a corresponding $\mathcal{FF}RAM$ $[\![\mathcal{R}]\!]$ such that the computation of $\mathcal{R}$ (starting with all registers empty) terminates if and only if $[\![\mathcal{R}]\!]$ has an infinite computation (starting with all registers empty).*

Proof. Given a RAM $\mathcal{R}$ with instructions $I_1, \cdots, I_n$ (assuming $I_n = I_{halt}$) and registers $r_0, \cdots, r_m$, with $[\![\mathcal{R}]\!]$ we denote the $\mathcal{FF}$RAM composed of the registers $r_0, r_1, \cdots, r_m, r_{m+1}, r_{m+2}, r_{m+3}$ and of the following instructions:

$$
\begin{array}{lcll}
J_1 & = & Inc(r_{m+1}) & \\
J_2 & = & DecJump(r_{m+1}, 1) & \\
J_{3i} & = & DecJump(r_{m+1}, halt) & (\text{for } 1 \leq i < n) \\
J_{3i+1} & = & Inc(r_{m+2}) & (\text{for } 1 \leq i < n) \\
J_{3i+2} & = & \begin{cases} Inc(r_j) & \text{if } I_i = Inc(r_j) \\ DecJump(r_j, 3s) & \text{if } I_i = DecJump(r_j, s) \end{cases} & (\text{for } 1 \leq i < n) \\
J_{3n+2j} & = & DecJump(r_j, 3n + 2j + 2) & (\text{for } 1 \leq i \leq m) \\
J_{3n+2j+1} & = & DecJump(r_{m+3}, 3n + 2j) & (\text{for } 1 \leq i \leq m) \\
J_{3n+2m+2} & = & DecJump(r_{m+2}, 3) & \\
J_{3n+2m+3} & = & Inc(r_{m+1}) & \\
J_{3n+2m+4} & = & DecJump(r_{m+3}, 3n + 2m + 2) &
\end{array}
$$

We prove that the computation of $\mathcal{R}$ starting from the state $(I_1, r_0 = 0, \cdots, r_m = 0)$ terminates if and only if $[\![\mathcal{R}]\!]$ has an infinite computation starting from the state $(J_1, r_0 = 0, \cdots, r_{m+3} = 0)$.

We first consider the *only-if* part. We assume that the RAM $\mathcal{R}$ terminates after the execution of k steps. The corresponding $\mathcal{FF}$RAM $[\![\mathcal{R}]\!]$ has the following infinite computation which contains exactly k wrong jumps. The wrong jumps are all executed at the beginning of the compuation in order to introduce in r_{m+1} the value k. Then the computation proceeds simulating infinitely many times the computation of $\mathcal{R}$. Note that at the end of each simulation, all the registers $r_1, \cdots, r_m$ are emptied, and the value k (which is introduced in r_{m+2} during the computation, is moved back in r_{m+1}). Note also that the register r_{m+3} is always empty, and that it is simply tested for zero by instructions that must always perform a jump.

We now consider the *if* part. Assume that the $\mathcal{FF}$RAM $[\![\mathcal{R}]\!]$ has an infinite computation. This computation starts with k executions of the instructions J_1 and J_2. The loop between these two instructions cannot proceed indefinitely as it contains a wrong jump. At the end of this first phase, the register r_{m+1} contains k. Then the computation continues by simulating the behavior of the RAM $\mathcal{R}$, and before executing every instruction the register r_{m+1} is decremented and the register r_{m+2} is incremented. If (by contradiction) the register r_{m+1} becomes empty before completing the simulation of $\mathcal{R}$, the computation should block. Thus, the simulation completes before simulating k steps. After, all the registers $r_0, \cdots, r_m$ are emptied, the value k is reintroduced in r_{m+1}, and a new simulation is started. This part of the computation, i.e. simulation of $\mathcal{R}$ and subsequent reset of the registers, surely terminates because the simulation of $\mathcal{R}$ includes at most k steps, and the subsequent reset of the registers cannot proceed indefinitely. We can conclude that an infinite computation includes infinitely many simulations

of the computation of $\mathcal{R}$ and, as an $\mathcal{FF}$RAM can perform only finitely many wrong jumps, infinitely many of these simulations are correct. This implies that the RAM $\mathcal{R}$ terminates within k computation steps. □

5 Conclusion

In this paper we have investigated the decidability of termination problems in CGF, a process algebra proposed in [2] for the compositional description of chemical systems. In particular, we have proved that existential termination is decidable, probabilistic termination is undecidable, and universal termination is decidable under a purely nondeterministic interpretation of CGF while it turns to be undecidable under the stochastic semantics.

It is worth saying that similar results hold also for lossy channels: universal termination is decidable in lossy channels while it turns out to be undecidable in their *probabilistic* variant [1]. Nevertheless, the result on lossy channels is not comparable with ours. In fact, in CGF process communication is synchronous (in lossy channels synchronous communication is not admitted) while in the lossy channel model it is asynchronous through unbounded FIFO buffers (that cannot be directly encoded in CGF).

Acknowledgement. We would like to acknowledge M. Bravetti, D. Soloveichik, H. Wiklicky, E. Winfree, and the anonymous referees for their insightful comments on previous versions of this paper.

References

1. P. Abdulla, C. Baier, P. Iyer, and B. Jonsson. Reasoning about Probabilistic Lossy Channel Systems. In *Proc. of 11th International Conference on Concurrency Theory (Concur)*, volume 1877 of *LNCS*, pages 320–333, 2000.
2. L. Cardelli. On Process Rate Semantics. *Theoretical Computer Science*, in press, 2008. Available at http://dx.doi.org/10.1016/j.tcs.2007.11.012.
3. H. Carstensen. Decidability Questions for Fairness in Petri Nets. In *Proc. of 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 247 of *LNCS*, pages 396–407, 1987.
4. J. Esparza and M. Nielsen. Decidability Issues for Petri Nets, 1994. Technical report BRICS RS-94-8.
5. J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. Springer Verlag, 1976.
6. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
7. M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, 1967.
8. D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with Finite Stochastic Chemical Reaction Networks. *Natural Computing*, in press, 2008. Available at http://dx.doi.org/10.1007/s11047-008-9067-y.