On the Computational Power of Biochemistry

Luca Cardelli[1] and Gianluigi Zavattaro[2]

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

Abstract. We explore the computational power of biochemistry with respect to basic chemistry, identifying complexation as the basic mechanism that distinguishes the former from the latter. We use two process algebras, the Chemical Ground Form (CGF) which is equivalent to basic chemistry, and the Biochemical Ground Form (BGF) which is a minimalistic extension of CGF with primitives for complexation. We characterize an expressiveness gap: CGF is not Turing complete while BGF supports a finite precise encoding of Random Access Machines, a well-known Turing powerful formalism.

1 Introduction

In this paper we introduce a minimal process algebra that aims to capture the essential primitives of biochemistry. Biochemistry is obviously based on chemistry, and in principle one can always express the behavior of a biochemical system by a collection of chemical reactions. But there is a major practical problem with that approach: the collection of reactions for virtually all biochemical systems is an infinite one. For example, just to express the chemical reactions involved in linear polymerization, we need to have a different chemical species for each length n of polymer P_n, with reactions to grow the polymer: $P_n + M \to P_{n+1}$. While each polymer is finite, the set of possible polymerization reactions is infinite. Nature adopts a more modular solution: the act of joining two molecules is called *complexation*, and polymers are made by iteratively complexing monomers. Each monomer obeys a *finite* simple set of rules that leads to the formation of polymers of any length; therefore, it seems that there should be a finite way of describing such systems. One can start by writing pseudo-reactions like $P + M \to P : M$, where $P : M$ is meant to represent a P(olymer) molecule attached to an extra M(onomer), yielding a longer polymer. However, there are in general many possible ways (that is, many different patches on the surface of a molecule) by which one molecule can exclusively form a complex with other molecules, and soon one needs to describe the *interface* of each molecule. This situation, while not commonly found in basic chemistry, is particularly acute in biochemistry, where virtually all reactions are governed by enzymes and molecular machines, which are themselves often built by complexation, and which usually operate by complexing with their reactants.

The intuitive idea of a molecule as a stateful entity with a connectivity interface is now common. Notations have emerged from biology that use such an idea to describe large biochemical systems [9, 8]. Many formalized and computerized approaches are currently being developed, including: practical tools, where molecules are drawn as boxes with connecting lines [6]; graph-rewriting and term-rewriting systems where a molecular complex is represented as a graph or term, and a reaction is a graph or term rewrite [7, 5]; coding techniques in process algebra, where complexation can be expressed via some advanced features [16]; and finally, specialized process algebras where molecular interfaces and complexation are taken as primitive [15, 4]. All these approaches aim to find a descriptive framework that goes beyond simple chemical reactions, and that can be used to represent common biochemical situations finitely and modularly.

The aim of this paper is then to investigate the computational boundary between chemistry and biochemistry. That is: what is the intrinsic power of complexation that gives it the ability to represent finitely what would otherwise have an infinite representation? To clarify this issue we study two formal systems, which for easy comparison are both based on the notion of molecules as stateful entities with an interface. One system, the Chemical Ground Form (CGF) has been presented in [3]: it is equivalent to basic chemistry, and it does not include complexation. The other system, the Biochemical Ground Form (BGF) is proposed in this paper as a minimalistic extension of CGF with complexation. As already mentioned, many richer formalisms can represent complexation too, but they also include mechanisms that have no direct biological implementation. Our proposal is minimalistic in the following sense: it adds only two basic actions called *association* and *dissociation*. Association allows two molecules to form a complex, dissociation allows them to subsequently break such a complex. Between an association event involving two molecules and their subsequent dissociation, the two molecules can still freely interact with other molecules or among themselves. In other more expressive formalisms, see for instance the so-called *exchange reactions* in [5], it is possible to specify events that change the internal state of one molecule only if it is complexed with another one having a particular state.

The main contribution of this paper is the formalization of the following expressiveness gap between chemistry and biochemistry: the CGF is not Turing complete while its minimalistic extension BGF is already Turing complete. The results on the CGF are obtained by resorting to papers on the computational power of basic discrete chemistry. In particular, we refer to works by Magnasco [11] and Soloveichik et al. [17] to show that only infinite CGF representations could be sufficiently expressive to precisely model any Turing powerful formalism. On the contrary, we show (as an original result) a finite BGF representation of Random Access Machines [14], a well known register based Turing powerful formalism.

The paper is structured as follows. In Section 2 we give the definition of the CGF and we discuss its computational power. In Section 3 we introduce the BGF, the new notation that enriches the CGF with complexation. In Section 4 we prove that the new process algebra is Turing complete, and finally in Section 5 we give some concluding remarks.

2 Chemical Ground Form

In this section we give the definition of the Chemical Ground Form (CGF): the notation for the representation of chemical systems presented in [3]. We first informally recall the notation, then we give the formal syntax and semantics.

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.

There are three kinds of actions. Action $\tau_{(r)}$ indicates the possibility for a molecule to be engaged in a unary reaction. For instance, the definition $A = \tau_{(r)}; (B|C)$ is used to specify the possibility for one molecule of species A to 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 [13], 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 that produces a molecule of species B, or in a binary reaction with another molecule able to execute the complementary action $!a_{(r')}$. In the second case, the molecule of species A contributes to the reaction by producing a new molecule of species C.

We now present the formal definition of the syntax of the CGF.

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):

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

Given a CGF (E, P), we assume that for every variable X occurring in P or E, there is exactly one definition $X = M$ in E.

In the following, trailing $\mathbf{0}$ are usually 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. Therefore, if P is a solution, then $\mathbf{0}|P$, $P|\mathbf{0}$, and P are syntactically equal.

We consider the discrete state semantics for the CGF defined in [3] 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 of 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.

In order to define the LTG of a chemical ground form we need to introduce 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 $P^\dagger \backslash (m_1, \cdots, m_n)$ 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:*

Unary: $\langle \{m.X.i\} : P^\dagger \xrightarrow{r} T^\dagger \rangle$ *such that* $P^\dagger.m = X$ *and* $E.X.i = \tau_{(r)}; Q$ *and* $T = (P^\dagger \backslash m)|Q$*;*

Binary: $\langle \{m.X.i, n.Y.j\} : P^\dagger \xrightarrow{r} T^\dagger \rangle$ *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* $T = (P^\dagger \backslash m, n)|Q|R$*.*

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 of an LTG). *If Ψ is an LTG, then $|\Psi|$ is its 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:* $|\Psi| = \{P \overset{r}{\mapsto} Q \text{ s.t. } \exists \langle l : P \xrightarrow{r} Q \rangle \in \Psi \text{ with } P \neq Q, \text{ and } r = \sum r_i \text{ s.t. } \langle l_i : P \xrightarrow{r_i} Q \rangle \in \Psi\}$.

We conclude this section by discussing the expressive power of the CGF. First of all, we recall the equivalence result proved in [3] between the CGF and *discrete chemistry*, that is, the traditional stochastic model of chemical kinetics that describes interactions among integer numbers of molecules as CTMCs [12]. More precisely, it is proved that every discrete chemical model (with unary and binary reactions) has a semantically equivalent CGF, and vice versa. By semantic equivalence between a chemical discrete model and a CGF, we mean that the underlying CTMCs are isomorphic.

In [11], Magnasco shows how to represent in discrete chemistry the computational model of electronic digital computers, based on finite logical circuits with an unbounded memory. Using the translation in [3], we can obtain an equivalent model also in CGF. This is not sufficient to prove that CGF is Turing complete. In fact, the technique proposed by Magnasco requires the exploitation of new species every time a new memory location is needed during the computation. Thus, the CGF system corresponding to a computable function requiring unbounded memory should include an unbounded number of species, thus also an unbounded number of definitions. This is not admitted in the CGF syntax.

The question about Turing completeness of the CGF can be answered in the light of more recent results proved in [17] by Soloveichik et al. In fact, they prove that discrete chemistry is not expressive enough to precisely model any Turing complete formalism, because the problem of deciding whether a certain molecule could be produced in a given chemical system is decidable. Therefore, we can conclude that also the CGF is not Turing complete because, by contraposition, if the CGF were Turing complete, the translation from CGF to discrete chemistry in [3] would allow one to model in discrete chemistry a Turing complete formalism with a finite number of species. It is also possible to derive this result more directly by a connection between the CGF and decidable properties of Petri nets as shown, e.g., in [18].

3 Biochemical Ground Form

In this section we present a minimalistic process algebra for biochemistry, obtained by extending the CGF with association and dissociation. We call the new process algebra *biochemical ground form* (BGF). Following a similar proposal outlined in [2], we consider two additional pairs of complementary actions, $\&?a_{(r)}, \&!a_{(r)}$ for association and $\%?a_{(r)}, \%!a_{(r)}$ for dissociation. Before presenting the formal syntax and semantics of the new actions, we introduce them informally by means of examples. To simplify the notation, in the examples we abstract away from the stochastic rates, e.g., we write $\&?a$ instead of $\&?a_{(r)}$.

Example 1 (Linearly growing polymer). Each complexation event involves exactly two partners. We imagine that the partners have two complementary surface patches that can interlock. If c represents a surface shape (say, a paraboloid), then $!c$ indicates one of the two patches (say, the convex one) and $?c$ indicates the complementary patch (the concave one). Then, $\&!c$ is the action that presents the convex patch, and $\&?c$ is the action that presents the concave patch. When two such *association* actions meet, an actual complexation event can take place, joining the two complementary surfaces.

A linearly growing polymer could be represented as follows, using a seed S and a collection of equal monomers M. The seed starts the chain by presenting a concave patch $?c$: this is our initial, zero-length, polymer. Each monomer presents a convex patch $!c$, which can bind with an existing polymer on the complementary concave patch. After (and only after) such a binding, a bound monomer M' presents another concave patch $?c$, so that the polymer can keep growing. Both the seed and each monomer can have further behavior, S' and M''.

$$\begin{array}{rcl} S & = & \&?c; S' \\ M & = & \&!c; M' \\ M' & = & \&?c; M'' \end{array}$$

Each complexation event creates a unique bond between exactly the two molecules that are joined to each other. This bond needs to be represented somehow, to make sure that a molecule can bind with only one other molecule at a time on any given patch. We represent such a bond as a unique key k that is shared by the two complexed molecules (think of k as a fresh number, or as a fresh channel in π-calculus [13]). Such unique keys, and related information, are collected in the *association history* of each molecule. So, the first interaction of an S with an M, which initially have empty association histories ($\mathbf{0}$), proceeds as follows:

$$S_{\mathbf{0}} \mid M_{\mathbf{0}} \to S'_{\langle ?c,k1\rangle} \mid M'_{\langle !c,k1\rangle}$$

Interaction with a second monomer then introduces a second fresh key in the histories:

$$S_{\mathbf{0}} \mid M_{\mathbf{0}} \mid M_{\mathbf{0}} \to S'_{\langle ?c,k1\rangle} \mid M'_{\langle !c,k1\rangle} \mid M_{\mathbf{0}} \to S'_{\langle ?c,k1\rangle} \mid M''_{\langle ?c,k2\rangle::\langle !c,k1\rangle} \mid M'_{\langle !c,k2\rangle}$$

This mechanism of creation of fresh association keys is repeated every time a new association is created between a monomer and the subsequent one.

It is worth observing that, in any reachable configuration, we can reconstruct from the association histories who is bound to whom, and on what surface the bond was formed. Note that the description of the system is finite (3 reagents, S, M, M'), but that polymers of any length can be assembled (assuming the initial availability of a corresponding amount of monomers).

Example 2 (Branching polymer). After complexation, a molecule is still free to perform additional complexations or other interactions. That is, complexation places no restrictions on the behavior of the original molecules, except for the fact that new complexations cannot occur on surfaces that are already occupied, and that decomplexations must happen consistently with prior complexations (as we discuss shortly). To illustrate this freedom, let us modify the previous example and allow each bound monomer to offer a seed for growing a new polymer branch:

$$\begin{array}{rcl} S & = & \&?c; S' \\ M & = & \&!c; M' \\ M' & = & \&?c; M'' \\ M'' & = & \&?d; M''' \\ N & = & \&!d; N' \\ N' & = & \&?c; N'' \end{array}$$

Where an M'' can bind through the interface d to an adaptor molecule N, which then offers another c surface for branching.

Example 3 (Actin-like polymer). Decomplexation is the inverse of complexation, that is, two formerly joined molecules can dissociate. We indicate by $\%!c$ the attempt to dissociate from the convex side, and $\%?c$ the attempt to dissociate from the concave side. When two complexed molecules attempt complementary dissociations, an actual decomplexation event can take place. To illustrate this situation, we describe a different kind of linear polymer: one that can grow only at one end, and can shrink only at the other end. There are four molecular states for each monomer: M^f (free monomer), M^l (monomer bound on the left), M^r (monomer bound on the right), and M^b (monomer bound on both sides). Each monomer has a left convex surface and a complementary right concave surface. A polymer should associate (grow) only on the right and should dissociate (shrink) only on the left.

$$\begin{array}{lcl} M^f & = & \&!c; M^l \oplus \&?c; M^r \\ M^l & = & \%!c; M^f \oplus \&?c; M^b \\ M^r & = & \%?c; M^f \\ M^b & = & \%!c; M^r \end{array}$$

A free monomer M^f can either associate on the left convex surface and become bound on the left, or associate on the right concave surface and become bound on the right. A monomer M^l bound only on the left can either dissociate on the left (if allowed by its partner, which must in fact be an M^r in this case) and return free, or associate on the right (with an M^f) and become bound on both sides. A monomer M^r bound only on the right can only dissociate on the right: that is, a polymer cannot grow on the left. A monomer M^b bound on both sides can only dissociate on the left (with an M^r): that is, a polymer cannot shrink on the right or break in the middle. These rules cover also the base cases when a polymer of length 2 initially forms or finally dissolves.

A decomplexation should succeed only between a pair of molecules that were actually complexed in their past history, and this can be checked by inspecting the unique keys introduced during complexation. For example let us consider two M^f molecules that complex and then immediately decomplex:

$$M^f_{\mathbf{0}} \mid M^f_{\mathbf{0}} \rightarrow M^l_{\langle !c,k\rangle} | M^r_{\langle ?c,k\rangle} \rightarrow M^f_{\mathbf{0}} | M^f_{\mathbf{0}}$$

The second transition is allowed to happen because M^l offers $\%!c$, M^r offers the complementary $\%?c$, and the same key k appears in both association histories on the c interface (and with the correct convexity). As a consequence of decomplexation, the keys are removed from the histories.

Example 4 (Unbounded linearly growing and shrinking polymer). Recursive definitions of the species behavior allows us to specify systems in which an unbounded number of monomers can be created. We use this ability to specify a linearly growing polymer started by a seed, that can also shrink removing the last associated monomer, and for which there is no fixed maximal length. In order to produce an unbounded number of monomer we consider a *factory* species able to continuosly produce monomers:

$$
\begin{array}{lcl}
Fact & = & \tau; (M^f | Fact) \\
S & = & \&?c; S' \\
S' & = & \%?c; S \\
M^f & = & \&!c; M^l \\
M^l & = & \%!c; M^f \oplus \&?c; M^b \\
M^b & = & \%?c; M^l
\end{array}
$$

It is easy to see that each seed molecule of species S has the ability to start the creation of a polymer that can grow and shrink along one direction without any fixed bound to its maximal length. We will exploit this technique in the proof of Turing completeness of the biochemical ground form in order to model registers, i.e., data structures on which increment, decrement and test for zero operations can be executed. The intuition is that increments are modeled by means of the creation and association of a new monomer, decrements by means of the elimination of the last associated monomer, and test for zero simply by checking the availability of a molecule of species S (the seed becomes of species S' when associated to a monomer).

Almost all new ingredients of the BGF have been presented in the examples above. The unique additional aspect that requires discussion deals with *molecule splitting*, that is the possibility for one reactant to produce more than one molecule. We allow only molecules without complexations (i.e. with an empty association history) to split. In fact, if we admit the splitting of complexed molecules, we also need to extend the language to allow for the specification of the distribution of the associations among the produced molecules: this is possible but somewhat cumbersome. The restriction to splitting only uncomplexed molecules simplifies the notation without limiting the computational power of the calculus.

The complete syntax of the BGF is defined as follows.

Definition 4 (Biochemical Ground Form (BGF)). *Consider the following denumerable sets:* Species *ranged over by variables X, Y, X^1, X^2, $\cdots$,* Channels *ranged over by a, b, $\cdots$, a totally ordered set of* Association keys *ranged over by k, k', $\cdots$. Moreover, let r, s, $\cdots$ be rates (i.e. positive real numbers).*

The syntax of BGF is as follows:

$E ::= \mathbf{0} \mid X = M, E$	*Reagents*
$M ::= \mathbf{0} \mid \pi; P \oplus M$	*Molecule*
$P ::= \mathbf{0} \mid X \vert P$	*Product*
$\pi ::= \tau_{(r)} \mid ?a_{(r)} \mid !a_{(r)}$	*Internal, Input, Output actions*
$\mid \&?a_{(r)} \mid \&!a_{(r)}$	*Association actions*
$\mid \%?a_{(r)} \mid \%!a_{(r)}$	*Dissociation actions*
$S ::= \mathbf{0} \mid X_H \vert S$	*Solution*
$H ::= \mathbf{0} \mid \langle ?a, k\rangle :: H \mid \langle !a, k\rangle :: H$	*Association history*
$BGF ::= (E, S)$	*Reagents and initial Solution*

Given a BGF (E, S), we assume that for every variable X occurring in P or E, there is exactly one definition $X = M$ in E. Moreover, we assume that an association key k either does not occur in S or it occurs in exactly two associations $\langle ?a, k\rangle$ and $\langle !a, k\rangle$ (for some channel a) stored in the history of two distinct molecules.[1]

In the following, trailing $\mathbf{0}$ are usually omitted also in association histories: for instance, we denote $\langle ?a, k\rangle :: \mathbf{0}$ simply with $\langle ?a, k\rangle$. Following this simplification, we can use a product P to specify a corresponding solution S, including the same molecules as in P, each of which having an empty association history. Moreover, we consider :: also as an operator over the syntax of association histories: for instance, if H and H' are $\mathbf{0}$-terminated association histories, according to the syntax above, then $H :: H'$ means appending the two lists into a single $\mathbf{0}$-terminated list. Therefore, if H is an association history, then $\mathbf{0} :: H$, $H :: \mathbf{0}$, and H are syntactically equal.

The semantics of BGF is defined, analogously to the semantics of CGF, in terms of a CTMC obtained in two steps, first the definition of a Labeled Transition Graph (LTG), then the extraction of a CTMC from the LTG. The second step is obtained in the same way as described in Definition 3. Thus we simply have to introduce a new definition for the LTG.

Due to the presence of the association histories, we need to introduce a new normal form for solutions.

Definition 5 (Normalized solution). *For a solution S of a well formed BGF, we indicate with $S^\dagger$ the normalized form of S obtained by*

1. *sorting the molecules first lexicographically according to their species name,*
2. *then sorting the molecules of the same species according to their initial key (i.e. the key of the first association in the history) putting the molecules without an initial key (i.e. with an empty history) before those with an initial key,*

[1] In BGF, we do not admit self-complexation, i.e., the possibility for one molecule to associate with itself. Still, it is possible for complexed molecules to form cycles; e.g., circular polymers.

3. *and finally, if there are pairs of molecules of the same species with the same initial key k, put the molecule with association $\langle ?a, k\rangle$ before the molecule with association $\langle !a, k\rangle$.*

Note that normalized solutions are well defined because for each pair of syntactically different molecules X_H and $X'_{H'}$ occurring in a well formed BGF, it defines whether X_H should precede $X'_{H'}$, or the vice versa. In fact, the unique case in which this is not defined is when they are of the same species and they both have an empty association history, thus they are syntactically identical.

On normalized solutions $S^\dagger$, we use the usual notation: $S^\dagger.m$ denotes the m-th molecule in $S^\dagger$, with $S^\dagger\backslash(m_1, \cdots, m_n)$ we denote the solution obtained by removing from $S^\dagger$ the m_i-th molecule for each $i \in \{1, \cdots, n\}$. We use also the following notation on association histories: with $H \setminus \langle ?a, k\rangle$ (resp. $H \setminus \langle !a, k\rangle$) we denote the history obtained by removing from H the association $\langle ?a, k\rangle$ (resp, $\langle !a, k\rangle$).

We now describe how to produce a Labeled Transition Graph from the Biochemical Ground Form (E, S). As in the previous section, $Next(E, S)$ is a set of quadruples $\langle l : S^\dagger \xrightarrow{r} T^\dagger\rangle$.

Definition 6 (LTG of a BGF). *Given a product P, with $P_{\mathbf{0}}$ we denote the solution obtained adding the empty association history $\mathbf{0}$ to the molecules in P. Given the BGF (E, S), we define $Next(E, S)$ as the set containing the following kinds of labeled transitions:*

Unary: $\langle\{m.X.i\} : S^\dagger \xrightarrow{r} T^\dagger\rangle$ *such that* $S^\dagger.m = X_H$ *and* $E.X.i = \tau_{(r)}; P$ *and* $T = (S^\dagger\backslash m)|V$ *such that*
- *if* $H = \mathbf{0}$ *then* $V = P_{\mathbf{0}}$,
- *if* $H \neq \mathbf{0}$ *then* $P = X'$ *and* $V = X'_H$;

Binary: $\langle\{m.X.i, n.Y.j\} : S^\dagger \xrightarrow{r} T^\dagger\rangle$ *such that* $S^\dagger.m = X_H$ *and* $S^\dagger.n = Y_{H'}$ *and* $m \neq n$ *and* $E.X.i = ?a_{(r)}; P$ *and* $E.Y.j = !a_{(r)}; Q$ *and* $T = (S^\dagger\backslash m, n)|V$ *such that*
- *if* $H = \mathbf{0}$ *and* $H' = \mathbf{0}$ *then* $V = P_{\mathbf{0}}|Q_{\mathbf{0}}$,
- *if* $H = \mathbf{0}$ *and* $H' \neq \mathbf{0}$ *then* $Q = Y'$ *and* $V = P_{\mathbf{0}}|Y'_{H'}$,
- *if* $H \neq \mathbf{0}$ *and* $H' = \mathbf{0}$ *then* $P = X'$ *and* $V = X'_H|Q_{\mathbf{0}}$,
- *if* $H \neq \mathbf{0}$ *and* $H' \neq \mathbf{0}$ *then* $P = X'$ *and* $Q = Y'$ *and* $V = X'_H|Y'_{H'}$;

Complexation: $\langle\{m.X.i, n.Y.j\} : S^\dagger \xrightarrow{r} T^\dagger\rangle$ *such that* $S^\dagger.m = X_H$ *and* $S^\dagger.n = Y_{H'}$ *and* $m \neq n$ *and* $E.X.i = \&?a_{(r)}; X'$ *and* $E.Y.j = \&!a_{(r)}; Y'$ *and for each* k' *we have that* $\langle ?a, k'\rangle \notin H$ *and* $\langle !a, k'\rangle \notin H'$ *and* $T = (S^\dagger \setminus m, n)|X'_{\langle ?a,k\rangle::H}|Y'_{\langle !a,k\rangle::H}$ *where k is the smallest association key among those that do not appear in the association histories in* $S^\dagger$;

Decomplexation: $\langle\{m.X.i, n.Y.j\} : S^\dagger \xrightarrow{r} T^\dagger\rangle$ *such that* $S^\dagger.m = X_H$ *and* $S^\dagger.n = Y_{H'}$ *and* $m \neq n$ *and* $E.X.i = \%?a_{(r)}; X'$ *and* $E.Y.j = \%!a_{(r)}; Y'$ *and there exists k s.t.* $\langle ?a, k\rangle \in H$ *and* $\langle !a, k\rangle \in H'$ *and* $T = (S^\dagger\backslash m, n)|X'_{H\setminus\langle ?a,k\rangle}|$ $Y'_{H'\setminus\langle !a,k\rangle}$.

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

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

It is easy to see that the assumption at the end of the Definition 4, i.e. that an association key k either does not occur in the solution or it occurs in exactly two associations $\langle ?a, k\rangle$ and $\langle !a, k\rangle$ (for some channel a) stored in the history of two distinct molecules, is preserved by the labeled transition system. In fact, the unique rules able to modify the association histories (the last two items in the Definition 6), removes both instances of an association key or create two instances of a new key, respectively.

The restriction, that we already informally discussed, that complexed molecules cannot split follows from the fact that splitting is possible only in the first two items of the Definition 6, and only in case the association history of the splitting molecule is empty.

Finally, we define the semantics of a BGF (E, S) as $|LTG(E, S)|$, that is the CTMC obtained from the labeled transition graph $LTG(E, S)$ according to the technique presented in Definition 3.

4 Turing completeness of BGF

We prove that Biochemical Ground Form is Turing complete. This result allows us to conclude that the association and dissociation actions cannot be encoded in the CGF, because the addition of these mechanisms makes BGF strictly more expressive.

In order to prove that BGF is Turing complete, we show how to model Random Access Machines (RAMs) [14], a well known Turing powerful formalism based on registers containing nonnegative natural numbers. The registers are used by a program, that is a set of indexed instructions I_i of two possible kinds:

- $i : Inc(r_j)$ that increments the register r_j and then moves to the execution of the instruction with index $i + 1$ and
- $i : DecJump(r_j, s)$ 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 assume the existence of a special instruction I_{halt} corresponding to program termination.

In our encoding of RAMs, we use a simplified notation for BGF definitions in which actions can be written in sequence. For instance the definition $A = \pi_1; \pi_2; C$ is a shorthand for the two definitions $A = \pi_1; B$ and $B = \pi_2; C$. Moreover, we do not show the stochastic rates (r) of the actions as they are not relevant: the execution of a RAM encoding proceeds deterministically (there are no probabilistic choices governed by the rates) and the speed of the RAM simulation is not important.

The encoding considers one species I^i for each instruction I_i. The behavior of the molecules of species I^i is to update the registers according to the corresponding instruction I_i, and then produce one molecule of species I^j corresponding to the subsequent instruction to be executed.

Formally, the species corresponding to the instructions are defined as follows:

$$I^i = \begin{cases} !inc_j; ?ack; I^{i+1} & \text{if } I_i = i : Inc(r_j) \\ !dec_j; ?ack; I^{i+1} \oplus !zero_j; I^s & \text{if } I_i = i : DecJump(r_j, s) \\ \mathbf{0} & \text{if } I_i = I_{halt} \end{cases}$$

In Figure 1 we graphically depict the above definitions using a graph-like notation: each species is represented by one node, and a transition labeled with an action π represents the possibility for the molecules of the source species to perform the action π producing one molecule of the target species. In case of

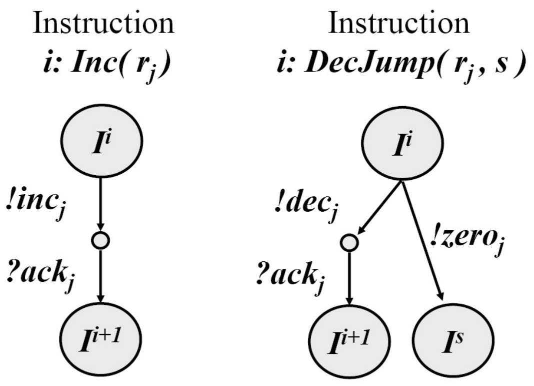

Fig. 1. Encoding of RAM instructions.

an increment instruction, a request for increment inc_j is considered, then an acknowledgment is required to have confirmation that the increment actually took place, and finally the next instruction is activated. In case of a decrement, either a decrement or a test for emptiness can take place: in the first case an acknowledgment is required before activating the next instruction; in the second case the jump is executed. In case of the terminating instruction, the corresponding molecule simply does nothing.

Each register r_j is modeled by a polymer similar to those described in the Example 4. In this case the seed is of species Z^j and the monomers are of species R^j. The number of monomers in the polymer coincides with the register content, namely, when the register holds the number l the polymer is composed of exactly l monomers. As it is not possible to know a priori the number of monomers necessary during the computation, we consider a factory, that is, a molecule of species RF^j which is responsible for the the generation of the molecules of species

R^j whenever they are needed. The last associated molecule in the polymer is the only one able to interact with the instruction molecules: if it is of species Z^j the active action is $?zero_j$, if it is of species R^j the active action is $?dec_j$. The effect of the execution of $?dec_j$ is the dissociation of the last associated molecule from the polymer.

The formal definition of the species used to model registers is as follows:

$$\begin{array}{lcl} Z^j & = & ?zero_j; Z^j \;\oplus\; \&?link_j; \%?link_j; Z^j \\ RF^j & = & ?inc_j; \big(RF^j | (\&!link_j; !ack; R^j)\big) \\ R^j & = & (\&?link_j; \%?link_j; R^j_{!link_j}) \;\oplus\; (?dec_j; \%!link_j; !ack; \mathbf{0}) \end{array}$$

In Figure 2 we graphically depict the encoding of registers. In this case we also have to represent the splitting of the molecules of species RF^j when performing the action $?inc_j$: the transition enters an intermediary *splitting* state represented with a bar, from which we have one outgoing transition for each of the produced molecules.

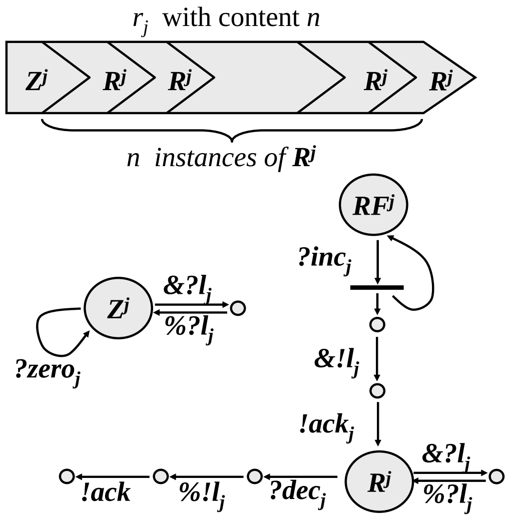

Fig. 2. Encoding of RAM registers.

The remainder of this section is devoted to the formal proof of correctness of this RAM encoding. We use the following notation. Given a RAM with registers $r_1, \cdots, r_n$, we denote with $(I_i, r_1 = l_1, \cdots, r_n = l_n) \mapsto (I_j, r_1 = l'_1, \cdots, r_n = l'_n)$ its possible steps of computation. Namely, if the RAM is going to execute instruction I_i, and the register contents are $l_1, \cdots, l_n$, respectively, then the next instruction is I_j and the new register contents are $l'_1, \cdots, l'_n$, respectively.

In the following we need to treat as equivalent some syntactically different solutions which represent the same biological system. For instance, the two solutions

$$Z^1_{\langle ?link_1,1\rangle} \mid R^1_{\langle ?link_1,4\rangle::\langle !link_1,1\rangle} \mid R^1_{\langle !link_1,4\rangle}$$
$$Z^1_{\langle ?link_1,2\rangle} \mid R^1_{\langle ?link_1,3\rangle::\langle !link_1,2\rangle} \mid R^1_{\langle !link_1,3\rangle}$$

both denote the polymer representing the register r_1 with content 2, even if they differ in their association keys. Formally, we have that two solutions S and T are *equivalent* if there exists an injective renaming ρ for the association keys in S such that $(S[\rho])^\dagger = T^\dagger$, where $S[\rho]$ denotes the result of the application of the injective renaming to the solution S. In the example above, the injective renaming used to prove that the two solutions are equivalent is $\{1 \mapsto 2, 4 \mapsto 3\}$.

We denote with $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$ the set of equivalent solutions which represent the RAM ready to execute the instruction I_i and in which the registers $r_1, \cdots, r_m$ have contents $l_1, \cdots, l_m$, respectively. Formally, $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$ is the set of solutions equivalent to:

$$\begin{array}{l} I^i \mid \\ RF^1 \mid Z^1_{\langle ?link_1,k_1^1\rangle} \mid R^1_{\langle ?link_1,k_1^2\rangle::\langle !link_1,k_1^1\rangle} \mid \cdots \mid R^1_{\langle !link_1,k_1^{l_1}\rangle} \mid \\ \cdots \\ RF^m \mid Z^m_{\langle ?link_m,k_m^1\rangle} \mid R^m_{\langle ?link_m,k_m^2\rangle::\langle !link_m,k_m^1\rangle} \mid \cdots \mid R^m_{\langle !link_m,k_m^{l_m}\rangle} \end{array}$$

Given a RAM denoted with $\mathcal{R}$, having instructions $I_1, \cdots, I_n$ and registers $r_1, \cdots, r_m$, we use $E_\mathcal{R}$ to denote the definitions of the species I^1, $\cdots$, I^n, Z^1, $\cdots$, Z_m, RF^1, $\cdots$, RF^m, and R^1, $\cdots$, R^m as defined above. Thus, given one of the possible configurations $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ of $\mathcal{R}$, we model it with the BGF $(E_\mathcal{R}, S)$ where S is any of the solutions in $[\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$.

We are now ready to prove the correctness result.

Theorem 1. *Let $\mathcal{R}$ be a RAM. Given one of its possible configurations $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ and a solution $S_0 \in [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$, we have that:*

- *either $I_i = I_{halt}$ and $Next(E_\mathcal{R}, S)$ is empty;*
- *or $(I_i, r_1 = l_1, \cdots, r_m = l_m) \mapsto (I_j, r_1 = l'_1, \cdots, r_m = l'_m)$ and there exist $S_1, \cdots, S_z$ such that for every $0 \leq x < z$ we have that $Next(E_\mathcal{R}, S_x^\dagger)$ contains only one transition which has $S_{x+1}^\dagger$ as its target state, and moreover $S_z \in [\![(I_j, r_1 = l'_1, \cdots, r_m = l'_m)]\!]$.*

Proof (outline). The proof is by case analysis on the following four possible cases: $I_i = I_{halt}$, $I_i = i : Inc(r_j)$, $I_i = i : DecJump(r_j, s)$ with $l_j > 0$, and $I_i = i : DecJump(r_j, s)$ with $l_j = 0$.

As a corollary of the theorem above, we have that if $\mathcal{R}$ is a RAM, given one of its possible configurations $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ and a solution $S \in [\![(I_i, r_1 = l_1, \cdots, r_m = l_m)]\!]$, there exists a solution containing the molecule I^{halt}

in $|LTG(E_{\mathcal{R}}, S)|$ if and only if the computation starting from the configuration $(I_i, r_1 = l_1, \cdots, r_m = l_m)$ halts. As the halting problem is undecidable for RAMs, we have that in the BGF the problem of deciding whether a certain molecule could be produced in a given system is undecidable. We have already observed that, on the contrary, this property is decidable for the process algebra CGF.

5 Conclusion

Turing-powerful mechanisms are not a requirement for building sophisticated nano-machines. Yet, the existence of Turing-powerful mechanisms guarantees a certain level of generality and flexibility in constructing machinery of any desired complexity, and provides evolution with adaptable toolkits to build upon. This paper highlights the fact that nature widely employs Turing-powerful mechanisms at the molecular level, and that it does so in a finitary combinatorial way that is qualitatively different from the common notion of chemical reactions between simple species. We have shown that the biochemical operations of complexation and decomplexation, formalized in a very basic form, are sufficient to raise expressiveness to the level of Turing-completeness, while simple chemistry (with finite descriptions) is not sufficient. In other words, finite programming constructs that are Turing powerful can be found in biochemistry but not in simple chemistry.

It is interesting to note that similar computational boundaries have been proved also in the context of process calculi based on membrane interactions such as endocytosis, exocytosis, fusion, and fission. In [1], Busi and Gorrieri prove that a basic process calculus including endoctytosis and exocytosis is Turing complete, while this is not the case when only fusion and fission are considered. This because endocytosis allows for the nesting of membranes with an unbounded depth, while this is not possible when only fission and fusion are considered. In BGF, instead of using membrane nesting, we consider a more basic complexation mechanism in order to generate structures with unbounded length.

The boundary of Turing-completeness gets even more interesting at the quantitative, approximate, level. For instance, recent work by Liekens and Fernando [10] shows how to approximate in discrete chemistry finite computations of Register Machines with an error probability smaller than any given precision $\delta > 0$. Soloveichik et al. [17], besides proving that in discrete chemistry it is not possible to precisely model any Turing powerful formalism (the result we have used in Section 2 to motivate that CGF is not Turing complete), show also how to approximate unbounded computations. A consequence of their results is that it is always decidable whether a certain molecule *could* be produced in a chemical system, while the question whether the system is *likely* to produce that molecule is in general undecidable. This opens interesting questions about what is actually decidable and what is undecidable in discrete chemistry. Some results recently proved along this line of research can be found in [18].

References

1. N. Busi and R. Gorrieri. On the Computational Power of Brane Calculi. In *Transactions on Computational Systems Biology*, volume 4220 of *LNCS*, pages 16–43. Springer, 2006.
2. L. Cardelli. Artificial Biochemistry. In *Proc. of Algorithmic Bioprocesses*, volume to appear of *LNCS*, 2008. Available at: http://lucacardelli.name.
3. 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.
4. L. Cardelli and S. Pradalier. Where Membranes Meet Complexes. In *Proc. of Concurrent Models in Molecular Biology (BioConcur05)*, 2005.
5. A. Credi, M. Garavelli, C. Laneve, S. Pradalier, S. Silvi, and G. Zavattaro. Modelization and Simulation of Nano Devices in nano-kappa Calculus. In *Proc. of Computational Methods in Systems Biology (CMSB07)*, volume 4695 of *LNCS*, pages 168–183, 2007.
6. V. Danos, J. Feret, W. Fontana, and J. Krivine. Kappa Factory, 2007. Available at: http://www.lix.polytechnique.fr/∼krivine/kappaFactory.html.
7. V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.
8. H. Kitano, A. Funahashi, Y. Matsuoka, and K. Oda. Using process diagrams for the graphical representation of biological networks. *Nature Biotechnolgy*, 23:961–966, 2005.
9. K.W. Kohn, M.I. Aladjem, J.N. Weinstein, and Y. Pommier. Molecular interaction maps of bioregulatory networks: a general rubric for systems biology. *Molecular biology of the cell*, 17(1):1–13, 2006.
10. A.M.L. Liekens and C.T. Fernando. Turing complete catalytic particle computers. In *Proc. of 9th European Conference on Artificial Life (ECAL07)*, volume 4648 of *Lecture Notes in Computer Science*, pages 1202–1211, 2007.
11. M.O. Magnasco. Chemical Kinetics is Turing Universal. *Physical Review Letters*, 78:1190–1193, 1997.
12. D.A. McQuarrie. Stochastic approach to chemical kinetics. *Journal of Applied Probability*, 4:413–478, 1967.
13. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
14. M. L. Minsky. *Computation: finite and infinite machines*. Prentice-Hall, Englewood Cliffs, 1967.
15. C. Priami and P. Quaglia. Beta Binders for Biological Interactions. In *Proc. of Computational Methods in Systems Biology (CMSB04)*, volume 3082 of *Lecture Notes in Computer Science*, pages 20–33, 2004.
16. C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80:25–31, 2001.
17. 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.
18. G. Zavattaro and L. Cardelli. Termination Problems in Chemical Kinetics, 2008. Available at: http://lucacardelli.name.