Computing with Genetic Gates, Proteins and Membranes

Nadia Busi - DSI, Universita' di Bologna, Italy

C. Zandron - DISCo, Universita' di Milano - Bicocca, Italy

Genetic P Systems

- Goal: to define systems which mimic the functioning of the genes
- A substance is produced when a gene is activated, that is
  - *Activator* substances are present
  - *Inhibitor* substances are absent
- Production of objects does not require that one or more objects are consumed
- Objects can disappear due to a decay process

Genetic P Systems

- We also consider
  - *Bind and release* rules, to simulate the action of protein channels on membranes
  - *Repressor rules* where some objects connect to other objects to block their action

Genetic gates

- $u_{act}, \neg u_{inh} :\longrightarrow (b, t)$ where $u_{act} \cap u_{inh} = \emptyset$.
  - $u_{act} \subseteq V$: *positive regulation* (activation).
  - Notice: we use sets of activators, as a genetic gate is never activated by more than one instance of the same protein
  - $u_{inh} \subseteq V$: *negative regulation (inhibition)*
  - $b \in V$: *transcription* of the gate (generated object - usually the expression of a genetic gate consists of a single protein)
  - $t \in I\!N \cup \infty$: *duration* of object b;

Genetic gates

- We also use $a, b, \neg c :\longrightarrow (c, 5)$ to denote the gate $\{a, b\}, \neg\{c\} :\longrightarrow (c, 5)$
- If either the activation or the inhibition is empty then we omit the corresponding set:
  $a :\longrightarrow (b, 3)$ corresponds to $\{a\}, \neg\emptyset :\longrightarrow (b, 3)$;
- The *nullary gate* $\emptyset, \neg\emptyset :\longrightarrow (b, 2)$ is written as $:\longrightarrow (b, 2)$.

Genetic gates functioning

- If one or more free inhibitor objects are present in the region where the gate is placed, then one of these objects (non-deterministically chosen) is bound to the gate, which cannot then be activated
- Gate $u_{act}, \neg u_{inh} :\rightarrow (b, t)$ is activated if the region it belongs to contains enough free activators and no free inhibitors. The activators in u_{act} are bound to the gate (and they cannot be used for activating any other gate in the evolution step).
- If the gate is activated, it performs the transcription and a new object (b, t) is produced.

Genetic gates - Object consumption

- Objects in u_{act} and u_{inh} **are not consumed** by the transcription operation: they will be released at the end of the operation.
- Each object starts with a *decay number*, which specifies the number of steps after which this object disappears.

Decay Number

- Is decreased after each evolution step;
- When it reaches the value zero, the object disappears;
- If the decay number of an object is ∞, then the object is persistent;
- Depends on the gate that produced the object (if the object is not present in the initial system), and not on the type of the object. Two occurrences of the same object produced by two different gates, may have different decay times.

Def Genetic P systems with timed degradation

GP system with timed degradation (of degree d, with $d \geq 1$):

$$\Pi = (V, \mu, w_1^0, \ldots, w_d^0, R_1, \ldots, R_d, i_0)$$

- V : finite alphabet;
- μ : *membrane structure*;
- w_i^0, $1 \leq i \leq d$: strings over $V \times (I\!N \cup \infty)$;
  multisets of objects of the form (a, t), where $a \in V$ and $t > 0$ is decay time;
- R_i, $1 \leq i \leq d$: finite **multisets** of *genetic gates* over V.
  Multisets (not sets): each rule can be used at most once in each step.
- i_0: *Output* membrane of Π.

Adding other types of rules

- Genetic gates alone is weak: no communication of objects is possible through the membranes
- Without communication, the system acts as a collection of separate systems, not as a whole
- We add two other types of rules, considering two different important cellular reactions:
- Bind&Release rules: mimics the communication of objects through a protein channel;
- Repressor rules: mimics the action substances that act to deactivate other substances in the cell;
- Obtained systems denoted by G^+P systems

Bind&Release Rules

$$u[v] \rightarrow v[u]$$

$u, v \in V^*$, s.t. $|uv| > 0$

- Two (multisets of) substances are *bound* to both side of a membrane
- They pass in opposite directions through the membrane itself, exchanging their position
- They are *released* in their (new) region
- *Weight* of a Bind and Release rule: $|u| + |v|$

Repressor Rules

$$a, b \rightarrow a\&b$$

where $a, b \in V$

- A repressor get in contact with another substance, creating a bond which cannot be destroyed;
- The created complex cannot be used anymore for any other reaction;
- Notice: repressors can create such bonds in any region of the cell. Hence, the set of repressor rules are valid for the whole system (i.e., not different sets of repressor rules, one for each region).

Result for G^+P Systems

Theorem: G^+P Systems, with Bind and Release rules of weight one, are Turing equivalent

Result proved by showing equivalence with Random Access Machines (RAMs), formalisms composed by

- n registers $r_1, \ldots, r_n$, that can hold arbitrary large natural numbers
- A sequence of indexed instructions $(1 : I_1), \ldots, (m : I_m)$.
- The execution stops when an instruction number higher than the length of the program is reached
- Output: contents of register 1 when the computation stops

RAM Instructions

Two types of instructions suffice:

- $(i : Succ(r_j))$: adds 1 to the contents of register r_j and goes to the next instruction;
- $(i : DecJump(r_j, s))$: if the contents of the register r_j is not zero, then decreases it by 1 and goes to the next instruction, otherwise jumps to the instruction s.

Construction of G^+P system

- Membrane structure: external membrane containing n children membranes, one for each register
- Value c_i in register r_i >>> c_i copies of object (r_i, ∞) in membrane i
- Instructions encoded by genetic gates
- Program counter with value i >>> An object (p_i, t) is present in the system (and no other object (p_j, t) with $j \neq i$ is present in the system).
  Initially, $(p_i, 2)$ is in the external membrane. All the objects representing the PC will be produced with duration 2
- Output: number of occurrences of object (r_1, ∞) in membrane 1.

Problem: termination

- Not trivial to define a halting configuration
- E.g.: system with a negative gate $\neg a :\rightarrow (b, t)$, reaching a configuration containing only a persistent object (a, ∞)
  - The gate "is working": at each step it inhibits the production of (b, t)
  - In fact, no real computation is performed...
- Termination condition used: like automata with final states - A configuration is reached which contains a persistent object (end, ∞) in the external membrane

Simulation of RAM

We provide RAM encoding s.t.

RAM terminates with output k

IFF

Corresponding G^+P system reaches a configuration where (end, ∞) is in membrane 0, and exactly k occurrences of (r_1, ∞) are in membrane 1

Ideas behind the encoding - Increment Instruction

To simulate the instruction $(i : Succ(r_j))$, the following instructions are executed in membrane 0:

- step 1: $p_i, \neg r_j :\rightarrow (r_j, \infty)$
- step 2: $p_i, r_j, \neg p_{i+1} :\rightarrow (p_{i+1}, 2)$
- step 3: $r_j[\ ]_j \rightarrow [r_j]_j$

If r_j enters membrane j before the object p_{i+1} is created, no new program counter $i+1$ will be created: the system will either stop in a failed computation or diverge without reaching a configuration with object *end*.

As the program counters have duration equal to 2, at step 3 the object p_i decays.

Ideas behind the encoding - Decrement instruction

To simulate the instruction $(i : DecJump(r_j, s))$ when $r_j = 0$, the following sequence of rules is executed:

- step 1: $p_i[\ ]_{l_j} :\rightarrow [p_i]_{l_j}$ (in membrane j)
- step 2: $p_i, \neg r_j :\rightarrow (p_s, 3)$ (in membrane j)
- step 3: $[p_s]_j \rightarrow p_s\ [\ ]_j$ (in membrane j)

After step 2 the object p_i decays.

If object p_i erroneously exits the membrane, then p_i decays just after exiting, and the system reaches a failed computation (or will diverge).

Ideas behind the encoding - Decrement instruction

To simulate the instruction $(i : DecJump(r_j, s))$ when $r_j > 0$:

- step 1: $p_i[\ ]_{l_j} :\longrightarrow [p_i]_{l_j}$ (in membrane j)
- step 2: $p_i, r_j, \neg dec_{i,j} :\longrightarrow decr_{i,j}$ (in membrane j)
- step 3: $r_j, decr_{i,j} \longrightarrow (p_{i+1}, 3)$ (in membrane j)
- step 4(1): $r_j, decr_{i,j} \longrightarrow r_j \& decr_{i,j}$ (in membrane j)
- step 4(2): $[p_{i+1}]_{l_j} \longrightarrow p_{i+1}\ [\ ]_{l_j}$ (in membrane j)

After step 2 the object p_i decays.

Steps 4(1) and 4(2) are executed in the same maximal parallelism step.

If the rule at step 4(1) takes place before step 3 (i.e., the repressor bounds to r_j before that p_{i+1} is created), then failure

Comments

- If some registers contain values greater than zero when RAM terminates, then the system reaches a configuration containing the *end* object, but because of gates $p_i, \neg r_j :\rightarrow (p_s, 3)$ the system will never terminate
- To guarantees that configurations containing the *end* object can perform no further computation, we could add to the RAM a further register r_{n+1} that is never decreased, and consider RAMs that terminate with all registers empty but r_{n+1}
- If an "erroneous" action is performed in the encoded system, then a failed configuration is reached (configuration without the *end* object). It is possible to produce an encoding that diverges when erroneous actions are performed: add to membrane 0 the gate $\neg end :\rightarrow loop$

Conclusions 1

- We use Bind and Release rules of weight 1. Symport rules of weight 2 (or antiport rules with one object entering the membrane and one object exiting the membrane) are already universal, without taking into account genetic gates
- Conjecture: with G^+P systems with only positive gates and with Bind and Release rules of weight 1 (and without repressor rules) it is possible to decide if a system can reach a configuration containing a *end* object.
- Conjecture: consider systems with positive and negative gates, and persistent objects only (no bind&release rules, nor repressor rules). The set of configurations of the system (maximal parallelism rule application) is a finite state machine: most of the behavioural properties can be decided

Conclusions 2

- Research topic: study different termination notions for GP systems
- Research topic: investigate questions already investigated for "classic" P systems in this framework, such as decidability, computational power, comparison with other formalism, etc.
- Research topic: apply genetic P systems to the systems biology area, to simulate various biological cell processes