Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Biologicky motivované výpočtové modely

(Dizertačná práca)

Michal Kováč

Vedúci: doc. RNDr. Damas Gruska, PhD. Bratislava, 2011

Čestne prehlasujem, že som túto dizertačnú prácu vypracoval samostatne s použitím citovaných zdrojov.

....

Poďakovanie

Osobitná vďaka patrí vedúcemu diplomovej práce doc. RNDr. Damasovi Gruskovi PhD. za cenné rady, námety, podnetné pripomienky a všestrannú pomoc, ktorú si hlboko vážim. Len vďaka mnohým prínosným konzultáciam a intenzívnej spolupráci som bol schopný napísať toto dielo. Nesmiem zabudnúť ani na RNDr. Branislava Rovana, CSc. a spolužiakov za to, že si na dizertačnom seminári našli čas, aby si vypočuli moju prezentáciu dizertačnej práce. Ďalšie poďakovania venujem rodičom a známym, ktorí to so mnou dokázali vydržať posledné týždne pred odovzdaním.

Abstrakt

Autor: Michal Kováč
Názov diplomovej práce: Biologicky motivované výpočtové modely
Škola: Univerzita Komenského v Bratislave
Fakulta: Fakulta matematiky, fyziky a informatiky
Katedra: Katedra informatiky
Vedúci bakalárskej práce: doc. RNDr. Damas Gruska, PhD.
Bratislava, 2011

Práca začína definovaním základných pojmov a končí interaktívnou prílohou.

Kľúčové slová: Mariáš, Teória hier, Minimax, Neúplna informácia.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

About computational models inspired by biology. Neural networks, evolution algorithms, membrane systems.

V teoretickej informatike je veľa oblastí, ktoré sú motivované inými vednými disciplínami. Veľkú skupinu tvoria modely motivované biológiou. Patria sem napríklad neurónové siete, výpočtové modely založené na DNA, evolučné algoritmy, ktoré si už našli svoje významné uplatnenie v informatike a dokázali, že sa oplatí inšpirovať biológiou. L-systémy sú špecializované na popisovanie rastu rastlín, ale našli si uplatnenie aj v počítačovej grafike, konkrétne vo fraktálnej geometrii. Ďalšie rozvíjajúce sa oblasti ešte čakajú na svoje významnejšie uplatnenie.

Jednou z nich sú membránové systémy. Je pomerne mladá oblasť - prvý článok bol publikovaný v roku 2000 (see [**?**])

Kapitola 1

Preliminaries

1.1 Formal languages theory

Our study is based on the classical theory of formal languages. We will recall some definitions:

Definition 1.1.1 *An* **alphabet** *is a finite nonempty set of symbols. Usually it is denoted by* Σ *or* V*.*

Definition 1.1.2 *A* **string** *over an alphabet is a finite sequence of symbols from alphabet.*

We denote by V^* the set of all strings over an alphabet V. By $V^+ = V^* - \{\varepsilon\}$ we denote the set of all nonempty strings over V.

Definition 1.1.3 *A* **language** *over the alphabet* V *is any subset of* V^**.*

1.2 Register machines

Definition 1.2.1 *A* n**-register machine** *is a tuple* $M = (n, P, i, h)$*, where:*

- n *is the number of registers,*
- P *is a set of labeled instructions of the form* $j : (op(r), k, l)$*, where* $op(r)$ *is an operation on register* r *of* M*, and* j*,* k*,* l *are labels from the set* $Lab(M)$ *(which numbers the instructions in a one-to-one manner),*

- *i is the initial label, and*
- *h is the final label.*

The machine is capable of the following instructions:

- $(add(r), k, l)$: Add one to the contents of register r and proceed to instruction k or to instruction l; in the deterministic variants usually considered in the literature we demand $k = l$.
- $(sub(r), k, l)$: If register r is not empty, then subtract one from its contents and go to instruction k, otherwise proceed to instruction l.
- *halt* : This instruction stops the machine. This additional instruction can only be assigned to the final label h.

A deterministic m-register machine can analyze an input $(n_1, \ldots, n_m) \in N_0^m$ in registers 1 to m, which is recognized if the register machine finally stops by the halt instruction with all its registers being empty (this last requirement is not necessary). If the machine does not halt, the analysis was not successful.

1.3 Multisets

Definition 1.3.1 *A multiset over a set X is a mapping $M : X \to \mathbb{N}$.*

We denote by $M(x), x \in X$ the multiplicity of x in the multiset M.

Definition 1.3.2 *The* **support** *of a multiset M is the set $supp(M) = \{x \in X | M(x) \geq 1\}$.*

It is the set of items with at least one occurence.

Definition 1.3.3 *A multiset is* **empty** *when it's support is empty.*

A multiset M with finite support $X = \{x_1, x_2, \ldots, x_n\}$ can be represented by the string $x_1^{M(x_1)} x_2^{M(x_2)} \ldots x_n^{M(x_n)}$.

Definition 1.3.4 *Multiset inclusion. We say that mutiset M_1 is included in multiset M_2 if $M_1(x) \leq M_2(x) \forall x \in X$. We denote it by $M_1 \subseteq M_2$.*

Definition 1.3.5 *The* **union** *of two multisets* $M_1 \cup M_2 : X \to \mathbb{N}$ *is defined as* $(M_1 \cup M_2)(x) = M_1(x) + M_2(x)$.

Definition 1.3.6 *The* **difference** *of two multisets* $M_1 - M_2 : X \to \mathbb{N}$ *is defined as* $(M_1 - M_2)(x) = M_1(x) - M_2(x)$.

Definition 1.3.7 *Product of multiset* M *with natural number* $n \in \mathbb{N}$ *is* $(n \cdot M)(x) = n \cdot M(x)$.

1.4 Multiset languages

The number of occurences of a given symbol $a \in V$ in the string $w \in V^*$ is denoted by $|w|_a$.

Definition 1.4.1 $\Psi_V(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$ *is called a Parikh vector associated with the string* $w \in V^*$, *where* $V = \{a_1, a_2, \ldots a_n\}$.

Definition 1.4.2 *For a language* $L \subseteq V^*$, $\Psi_V(L) = \{\Psi_V(w) | w \in L\}$ *is the Parikh mapping associated with* V.

Example 1.4.1 Consider an alphabet $V = \{a, b\}$ and a language $L = \{a, ab, ba\}$. $\Psi_V(L) = \{(1, 0), (1, 1)\}$. Notice that Parikh image of L has only 2 element while L has 3 elements.

Definition 1.4.3 *If FL is a family of languages, by PsFL is denoted the family of Parikh images of languages in FL.*

Parikh images

Kapitola 2

Membrane computing

Membranes intro

Natural computing is a recent field of research wchi tries to imitate nature in the way it ”computes”, learning new computing models and computing paradigms experimented for billions of years by nature.

Neural networks, genetic algorithms and DNA computing are already well established research fields.

However, nature computes not only at the neural or genetic level, but also at the cellular level. In general, any non-trivial biological system has a hierarchical structure where objects and information flows between regions, what can be interpreted as a computation process.

The regions are typically delimited by various types of membranes at different levels from cell membranes, through skin membrane to virtual membranes which delimits different parts of an ecosystem. This hierarchical system can be seen in other field such as distributed computing, where again well delimited computing units coexist and are hierarchically arranged in complex systems from single processors to the internet.

Membranes keep together certain chemicals or information and selectively determines which of them may pass through.

From these observations, Paun [1] introduces the notion of a membrane structure as a mathematical representation of hierarchical architectures composed of membranes. It is usually represented as a Venn diagram with all the considered sets being subsets of a unique set and not allowed to be intersected. Every two sets are either one the subset of the other, or disjoint.

P systems

Kapitola 3

P systems

In previous chapter we introduced the notions of membrane and membrane structure.

The next step is to place certain objects in the regions delimited by the membranes. The objects are identified by their names, mathematically symbols from a given alphabet.

Several copies of the same object can appear in a region, so we will work with multisets of objects.

In order to obtain a computing device, we will allow the objects to evolve according to evolution rules. Any object, alone or together with another objects, can be transformed in other objects, can pass through a membrane, and can dissolve the membrane in which it is placed.

All objects evolve at the same time, in parallel manner across all membranes.

The evolution rules are hierarchizes by a priority relation, which is a partial order.

These aspects all together forms a P system as introduced in [1].

In section 3.1 we will provide formal definition of a P system.

3.1 Definitions

P system is a tuple $(V, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m)$, where:

- V is the alphabet of symbols,

- μ is a membrane structure consisting of m membranes labeled with numbers $1, 2, \ldots, m$,
- $w_1, w_2, \ldots w_m$ are multisets of symbols present in the regions $1, 2, \ldots, m$ of the membrane structure,
- $R_1, R_2, \ldots R_m$ are finite sets of the rewriting rules asssociated with the regions $1, 2, \ldots, m$ of the membrane structure.

Each rewriting rule may specify for each symbol on the right side, whether it stays in the current region, moves through the membrane to the parent region or through membrane to one of the child regions. An example of such rule is the following: $abb \rightarrow (a, here)(b, in)(c, out)(c, here)$.

A **configuration** of a P system is represented by it's membrane structure and the multisets of objects in the regions.

A **computation step** of P system is a relation $\Rightarrow$ on the set of configurations such that $C_1 \Rightarrow C_2$ iff:

For every region in C_1 (suppose it contains a multiset of objects w) the corresponding multiset in C_2 is the result of applying a multiset of maximal simultaneously applicable multiset rewriting rules in R_w^{msap} to w.

In other words, a maximal multiset of rules is applied in each region.

For example, let's have two regions with multisets aa and b. In the first region there is a rule $a \rightarrow b$ and in the second membrane there is a rule $b \rightarrow aa$. The only possible result of a computation step is bb, aa. The first rule was applied twice and the second rule once. No more object could be consumed by rewriting rules.

Computation of a P system consists of a sequence of steps. The step S_i is appied to result of previous step S_{i-1}. So when $S_i = (C_j, C_{j+1})$, $S_{i-1} = (C_{j-1}, C_j)$.

Result of computation is multiset of symbols that left the skin membrane in the configuration after the last computation step. For one initial configuration there can be multiple possible results. It follows from the fact that there exist more than one maximal multiset of rules that can be applied in each step.

P system defines a parikh image of a language: the set of possible results of computations.

TODO:

How is P system defined. Mostly taken from my article. Multiset rewriting. Accepting vs generating model, active or passive membranes, quality vs quantity aspects.

3.2 P system variants

We are interested in computation power of various variants of P systems. Especially those that are universal (Turing completeness).

3.2.1 Parallelism options

Maximal parallelism, minimal parallelism, n-parallelism, sequential models.

3.2.2 Contextivity rules

Context rules vs cooperational rules, catalytic rules, symmetric cooperational rules, catalytic rules, promoters, inhibitors, context-free rules.

3.2.3 Priority rules

3.2.4 Energy of membranes

3.2.5 Calculi of Looping Sequences

3.3 Case studies

Vultures in Pyrenees, Scavangers of Pyrenees.

Záver

Literatúra

[1] Gheorghe Păun. Computing with membranes. Technical Report 208, Turku Center for Computer Science-TUCS, 1998. (www.tucs.fi).

Štatistiky

Tu budú štatistiky.