Journal of Universal Computer Science, vol. 10, no. 5 (2004), 600-619
submitted: 1/4/04, accepted: 10/4/04, appeared: 28/5/04

Simulating the Fredkin Gate with Energy–Based P Systems

Alberto Leporati, Claudio Zandron, Giancarlo Mauri
(Università degli Studi di Milano – Bicocca, Italy
{leporati, zandron, mauri}@disco.unimib.it)

Abstract: Reversibility plays a fundamental role when the possibility to perform computations with minimal energy dissipation is considered. Many papers on reversible computation have appeared in literature, the most famous of which is certainly the work of Bennett on (universal) reversible Turing machines. Here we consider the work of Fredkin and Toffoli on conservative logic, which is a mathematical model that allows to describe computations which reflect some properties of microdynamical laws of physics, such as reversibility and conservation of the internal energy of the physical system used to perform the computations. The model is based upon the Fredkin gate, a reversible and "conservative" (according to a definition given by Fredkin and Toffoli) three–input/three–output boolean gate.

In this paper we introduce energy–based P systems as a parallel and distributed model of computation in which the amount of energy manipulated and/or consumed during computations is taken into account. Moreover, we show how energy–based P systems can be used to simulate the Fredkin gate. The proposed P systems that perform the simulations turn out to be themselves reversible and conservative.

Key Words: Energy–based P systems, Fredkin gate, conservative logic, reversibility, conservativeness

Category: F.1.1, F.4.2

1 Introduction

Considerations of thermodynamics of computing started in the early fifties of the twentieth century, when the possibility to perform computations with minimal energy dissipation was first considered, and continued in the following decades. As a result some bounds on the amount of dissipated energy during transmission and computation were established [Landauer 1961, Bennett 1973, Bekenstein 1981, Landauer 1982], and some quantum theoretic models of computation were proposed [Benioff 1982]. As shown in [Landauer 1961], erasing a bit necessarily dissipates $kT \ln 2$ Joule in a computer operating at temperature T, and generates a corresponding amount of entropy. Here k is Boltzmann's constant and T the absolute temperature in degrees Kelvin, so that $kT \approx 3 \times 10^{-21}$ Joule at room temperature. However, in [Landauer 1961] Landauer also demonstrated that only *logically irreversible* operations necessarily dissipate energy when performed by a physical computer. (An operation is *logically reversible* if its inputs can always be deduced from its outputs.) This result gave substance to the idea that logically reversible computations could be performed with zero internal energy dissipation. Indeed, since the appearance of [Landauer 1961] many authors have concentrated their attention on reversible computations. The importance of reversibility has grown further with the development of *quantum computing*, where the dynamical behavior of quantum systems is usually described by means of unitary operators, which are inherently logically reversible. Let us note, however, that computing in a logically reversible way says nothing about whether or not the computation dissipates energy: it merely means that the laws of physics do not require that such a dissipation occurs.

Many papers on reversible computation have appeared in literature; the most famous are certainly the work of Bennett on (universal) reversible Turing machines [Bennett 1973], and the work of Fredkin and Toffoli on conservative logic [Fredkin and Toffoli 1982]. In particular, conservative logic has been introduced as a mathematical model that allows one to describe computations which reflect some properties of microdynamical laws of physics, such as reversibility and conservation of the internal energy of the physical system used to perform the computations. In this model, computations are performed by reversible circuits composed by Fredkin gates.

In this paper we introduce energy–based P systems as a parallel and distributed model of computation in which the amount of energy manipulated and/or consumed during computations is taken into account. In the most general version, a given amount of energy is associated with each object, membrane and rule of the system. Some energy units are provided from the external environment and are used to build, transform or move objects. When an object is transformed into another object as the effect of the application of a rule, the required (resp., exceeding) energy is taken from (resp., released to) the region where the rule is applied. The application of each rule consumes a given amount of energy. Membranes can be thought of as energy reservoirs which are able to accumulate a (possibly bounded) amount of energy and subsequently release part of it. A special case of energy–based P systems are *conservative* P systems, where the amount of energy entering the system with the input values is completely returned with the output values at the end of the computation.

We show how the Fredkin gate can be simulated with energy–based P systems. The proposed P systems that perform the simulation turn out to be themselves reversible and conservative. The simulation of reversible Fredkin circuits is currently under examination and is proposed here as a direction for future work.

This is by no means the first time that energy is considered when dealing with P systems. We recall in particular [Alford 2002, Freund 2002, Freund et al. 2004, Păun et al. 2001, Frisco 2004, Frisco and Ji 2002a, Frisco and Ji 2002b]. The last two papers were inspired by [Ji 2001]. Moreover, this is not even the first paper which deals with the simulation of boolean gates and circuits by biologically inspired models of computation: for instance, in [Ogihara and Ray 1996] a model for simulating boolean circuits (composed by AND, OR and NOT gates) with DNA algorithms is proposed, in [Erk 1999] the same goal is reached using finite splicing, and in [Ceterchi and Sburlan 2003] some P systems that simulate boolean circuits are presented. In [Prakash and Krithivasan 2004], the ideas found in [Ceterchi and Sburlan 2003] are applied to tissue P systems. Finally we also mention [Klein et al. 1999], where a biomolecular implantation of logically reversible computation using short strands of DNA as input and output lines of a Fredkin gate is demonstrated, and a method to connect Fredkin gates in order to create more complicated genetic networks is described.

The paper is organized as follows. In section 2 we recall some basic notions on conservative logic and the Fredkin gate. In section 3 we introduce the basic version of energy–based P systems, where energy is only associated with symbol objects with the requirement that for each rule the amount of energy occurring on the left side is the same as the amount of energy occurring on the right side. Conservative energy–based P systems are also introduced. In section 4 we show how the Fredkin gate can be simulated using this kind of P systems. In section 5 we propose some extensions to our model together with some open problems. Section 6 concludes the paper with further directions for future research.

2 Conservative Logic and the Fredkin Gate

Conservative logic is a mathematical model of computation based upon the so called *Fredkin gate*, a three–input/three–output boolean gate originally introduced by Petri in [Petri 1965] whose input/output map $\text{FG} : \{0,1\}^3 \to \{0,1\}^3$ associates any input triple (x_1, x_2, x_3) to its corresponding output triple (y_1, y_2, y_3) as follows:

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= (\neg x_1 \wedge x_2) \vee (x_1 \wedge x_3) \\ y_3 &= (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3) \end{aligned} \tag{1}$$

Table 1 shows the truth table of the Fredkin gate. A useful point of view is that the Fredkin gate behaves as a *conditional switch* (see Fig. 1): that is, $\text{FG}(1, x_2, x_3) = (1, x_3, x_2)$ and $\text{FG}(0, x_2, x_3) = (0, x_2, x_3)$ for every $x_2, x_3 \in \{0,1\}$. In other words, x_1 can be considered as a control input whose value determines whether the input values x_2 and x_3 have to be exchanged or not.

The Fredkin gate is *functionally complete* for Boolean logic: in fact, by fixing $x_3 = 0$ we get $y_3 = x_1 \wedge x_2$, whereas by fixing $x_2 = 1$ and $x_3 = 0$ we get $y_2 = \neg x_1$.

The Fredkin gate is also *reversible*, that is, it computes a bijective map on $\{0,1\}^3$. As we can see in Table 1, for every input/output pair the number of 1's in the input triple is the same as the number of 1's in the output triple. In

x_1	x_2	x_3	$\mapsto$	y_1	y_2	y_3
0	0	0		0	0	0
0	0	1		0	0	1
0	1	0		0	1	0
0	1	1		0	1	1
1	0	0		1	0	0
1	0	1		1	1	0
1	1	0		1	0	1
1	1	1		1	1	1

Table 1: Truth table of the Fredkin gate

Figure 1: The Fredkin gate as a conditional switch

other words, the output triple is obtained by applying an appropriate permutation to the input triple. Let us note that the applied permutation is input–dependent: namely, if $x_1 = 1$ then the applied permutation is $(2\ 3)$, whereas if $x_1 = 0$ then the applied permutation is the identity. Indeed, FG seems to be the most elementary input–dependent permutation which can be conceived. In [Fredkin and Toffoli 1982] Fredkin and Toffoli interpret the conservation of the number of 1's between input and output triples as the conservation of the amount of energy associated with the input triple, thus assuming that two different triples having the same number of 0's and 1's require the same amount of energy to be realized in a physical system. Let us note that conservativeness is defined (both here and in [Fredkin and Toffoli 1982]) as a mathematical notion; namely, it is *not* required that the entire energy used to perform the computation is preserved, or that the computing device be a conservative *physical* system (an ideal but unrealistic situation). In particular, we do not consider the energy needed to actually *perform* the computation, that is, to supply the computing device.

Basing upon these observations, Fredkin and Toffoli introduce a computational model for reversible and conservative computations. Computations are performed by reversible Fredkin circuits having the same number n of input and output lines. Under this constraint, the conservativeness requirement (preservation of the number of 1's) is again equivalent to the requirement that the output n-tuple is obtained by applying an appropriate (input–dependent) permutation to the input n-tuple. Here we just mention the fact that every permutation can be written in a unique way (up to the order of factors) as a composition of transpositions. This means not only that the Fredkin gate can be used to build an appropriate circuit to perform any given conservative computation (and thus it is universal also in this sense with respect to conservative computations), but also that it is the most elementary conceivable operation that can be used to describe conservative computations.

It is important to note that reversibility and conservativeness are two independent notions: a function (computed by a gate or circuit) may be only reversible, only conservative, both or none of them. However, for any function $f : \{0,1\}^n \to \{0,1\}^m$ it is possible to build a new function $f_R : \{0,1\}^{n+m} \to \{0,1\}^{n+m}$ such that f_R is a bijection on the set $\{0,1\}^{n+m}$ and moreover:

$$\forall\, \underline{x} \in \{0,1\}^n \quad f_R(\underline{x}, \underline{0}_m) = (\underline{x}, f(\underline{x})),$$

where $\underline{0}_m$ is the m-tuple consisting of all 0's. The function f_R is simply defined as follows:

$$\forall\, \underline{x} \in \{0,1\}^n,\ \forall\, \underline{y} \in \{0,1\}^m \quad f_R(\underline{x}, \underline{y}) = (\underline{x}, \underline{y} \oplus f(\underline{x})),$$

where $\oplus$ denotes the bitwise XOR operation. Hence, given a circuit that computes the function f it is always possible to build a reversible circuit that, using some additional input and output lines, is able to compute the values assumed by f on its last m output lines.

Analogously, for any function $f : \{0,1\}^n \to \{0,1\}^m$ it is possible to build a conservative function f_C that computes the values assumed by f in its first m output bits. Precisely, let us define the following quantities:

$$O_f = \max\left\{0, \max_{\underline{x} \in \{0,1\}^n} \{E_m(f(\underline{x})) - E_n(\underline{x})\}\right\},$$
$$Z_f = \max\left\{0, \max_{\underline{x} \in \{0,1\}^n} \{E_n(\underline{x}) - E_m(f(\underline{x}))\}\right\}.$$

Informally, O_f (resp., Z_f) is the maximum number of 1's (resp., 0's) in the output pattern that should be converted to 0 (resp., 1) in order to make the function conservative. We can thus define f_C as an $(n + O_f + Z_f)$–input/$(m + O_f + Z_f)$–output function such that:

$$\forall\, \underline{x} \in \{0,1\}^n \quad f_C(\underline{x}, \underline{1}_{O_f}, \underline{0}_{Z_f}) = (f(\underline{x}), \underline{1}_{w(\underline{x})}, \underline{0}_{z(\underline{x})}),$$

where $\underline{1}_k$ (resp., $\underline{0}_k$) is the k–tuple consisting of all 1's (resp., 0's), and the pair $(\underline{1}_{w(\underline{x})}, \underline{0}_{z(\underline{x})}) \in \{0,1\}^{O_f+Z_f}$ is such that $w(\underline{x}) = O_f + E_n(\underline{x}) - E_m(f(\underline{x}))$

and $z(\underline{x}) = Z_f - E_n(\underline{x}) + E_m(f(\underline{x}))$. Hence, we use some additional inputs (resp., outputs) in order to provide (resp., remove) the required (resp., exceeding) energy that allows f_C to compute f in a conservative way.

It is also possible, for any given function $f : \{0,1\}^n \to \{0,1\}^m$, to extend the reversible function f_R built above to a reversible and conservative function f_{RC} by adding some additional input and output bits. For the proof we refer the reader to [Cattaneo et al. 2002b].

In [Cattaneo et al. 2002a, Cattaneo et al. 2002b, Mauri and Leporati 2003] conservativeness has been extended to reversible and non reversible gates whose input and output lines may assume a finite number d of truth values. Some many–valued extensions of the Fredkin gate have also been presented. By associating equispaced energy levels to the truth values, the authors have shown that their notion of conservativeness corresponds to the energy conservation principle applied to the data which are manipulated during the computation. In the same papers the notion of *conservative computation* has been introduced, under the reasonable assumption that a gate may *store*, or *accumulate*, some energy in its internal machinery. Moreover, a new NP–complete decision problem concerning conservative computations has been defined. Some constant factor approximation algorithms for an associated NP–hard optimization problem are currently under examination.

3 Energy–based P Systems

P systems (also called *membrane systems*) were introduced in [Păun 1998] as a new class of distributed and parallel computing devices, inspired by the structure and functioning of cells. The basic model consists of a hierarchical structure composed by several membranes, embedded into a main membrane called the *skin*. Membranes divide the Euclidean space into *regions*, that contain some *objects* (represented by symbols of an alphabet) and *evolution rules*. Using these rules, the objects may evolve and/or move from a region to a neighboring one. The rules are applied in a nondeterministic and maximally parallel way: all the objects that may evolve are forced to evolve. A *computation* starts from an initial configuration of the system and terminates when no evolution rule can be applied. The result of a computation is the multiset of objects contained into an *output membrane* or emitted from the skin of the system.

In what follows we assume that the reader is already familiar with the basic notions and the terminology underlying P systems. For details, and a systematic introduction to the subject, see the monography [Păun 2002]. The latest information about P systems can be found in [P systems Web page].

In order to take into account the amount of energy used during computations, we define a new model which we call *energy–based P system*. In this model, we consider a special symbol e which denotes a free energy unit floating into regions; moreover, the rules are defined according to conservativeness considerations. We will show how this model can be used to simulate the Fredkin gate.

Formally, an energy–based P system (of degree $m \geq 1$) is a construct

$$\Pi = (A, \varepsilon, \mu, e, w_1, \ldots, w_m, R_1, \ldots, R_m, i_{\text{in}}, i_{\text{out}}),$$

where:

- A is an alphabet; its elements are called *objects*;
- $\varepsilon : A \to \mathbb{R}^+$ is a linear mapping that associates to each object $a \in A$ the real value $\varepsilon(a)$ (also denoted by ε_a), which can be thought of as the "energy value of a". Precisely, if $A = \{a_1, a_2, \ldots, a_d\}$, then for all $i \in \{1, 2, \ldots, d\}$ it holds $\varepsilon(a_i) = \varepsilon(a_1) + (i-1)\delta$ for an appropriate real value $\delta > 0$. Hence, the energy values considered in the system are equispaced by the quantity δ. Through an appropriate rescaling, we can always assume that all energy values are positive integer values, and that $\delta = 1$;
- μ is a hierarchical membrane structure consisting of m membranes. For the sake of clarity, we will label membranes with mnemonic identifiers which recall their function;
- $e \notin A$ is a special symbol that denotes one *free energy* unit, that is, one unit of energy which is not embedded into any object;
- w_i, for all $i \in \{1, \ldots, m\}$, specify the multisets (over $A \cup \{e\}$) of objects initially present in region i;
- R_i, for all $i \in \{1, \ldots, m\}$, is a finite set of evolution rules over A associated with region i. Only rules of the following types are allowed:

  $$ae^k \to (b, p) \quad , \qquad a \to (b, p)e^k \quad , \qquad e \to (e, p),$$

  where $a, b \in A$, $p \in \{\text{here}, \text{in}(name), \text{out}\}$ and k is a non negative integer;
- i_{in} is an integer between 1 and m and specifies the input membrane of Π;
- i_{out} is an integer between 0 and m and specifies the output membrane of Π. If $i_{\text{out}} = 0$, then the environment is used for the output, that is, the output value is the multiset of objects (over A) emitted from the skin.

A special attention is due to the definition of rules. The meaning of rule $ae^k \to (b, p)$, with $a, b \in A$, $p \in \{\text{here}, \text{in}(name), \text{out}\}$, and k a positive integer number, is the following: the object a, in presence of k free energy units, is allowed to be transformed into object b. If $p = \text{here}$, then the new object b

remains in the same region; if $p = \text{out}$, then b exits from the current membrane. Finally, if $p = \text{in}(name)$, then b enters into the membrane labelled with $name$, which must be a child of the current membrane in the membrane hierarchy.

The meaning of rule $a \to (b,p)e^k$, when k is a positive integer number, is analogous. The object a is allowed to be transformed into object b by releasing k units of free energy. As above, the new object b may optionally move one level up or down into the membrane hierarchy. The k free energy units can now be used by another rule to produce "more energetic" objects from "less energetic" ones.

When $k = 0$ the rule $ae^k \to (b,p)$ is written as $a \to (a,p)$, and simply moves (if $p \neq \text{here}$) the object a upward or downward into the membrane hierarchy, without acquiring nor releasing any free energy unit. Analogously, rules $e \to (e,p)$ simply move (if $p \neq \text{here}$) one unit of free energy upward or downward into the membrane hierarchy.

A further constraint for the definition of rules is that each rule must be "conservative", in the sense that the amount of energy occurring on the left side of the rule must be the same as the amount of energy which occurs on the right side.

With a little abuse of notation, when the pair (x,p), with $x \in A \cup \{e\}$ and $p \in \{\text{here}, \text{in}(name), \text{out}\}$, appears into a rule we will write x_p. Also, if $p = \text{in}(name)$ and no confusion arises we will usually write just the name of the membrane. Moreover, instead of writing e^k we will sometimes explicitly write k instances of e. It is also understood that the position of e^k (that is, on the left or on the right of the symbol of A) either into the left or into the right side of a rule is uninfluent. Finally, when the position p of an object which occurs in the right side of a rule is "here" we will omit to write it.

Example 1. Let us assume $A = \{a,b,c,d\}$, where the objects have energy values $\varepsilon_a = 1$, $\varepsilon_b = 2$, $\varepsilon_c = 3$ and $\varepsilon_d = 4$. Then the rule $be^2 \to (d, \text{out})$ (also written as $bee \to d_{\text{out}}$) transforms an instance of the object b into an instance of the object d, provided that two free energy units are available, and makes the new object d leave the current membrane.

On the other hand, the rule $c \to (a, \text{here})e^2$ (also written as $c \to aee$) transforms an instance of the object c into an instance of the object a and releases two free energy units into the region in which the rule is defined.

A *configuration* of Π is the collection $\{M_1, \ldots, M_m\}$ of multisets (over $A \cup \{e\}$) of objects contained in each region of the system. $\{w_1, \ldots, w_m\}$ is called the *initial configuration*. For two configurations $\{M_1, \ldots, M_m\}$, $\{M'_1, \ldots, M'_m\}$ of Π we write $\{M_1, \ldots, M_m\} \Rightarrow \{M'_1, \ldots, M'_m\}$ to denote a *transition* from $\{M_1, \ldots, M_m\}$ to $\{M'_1, \ldots, M'_m\}$, that is, the parallel application of one or more rules of the system. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$. A *final configuration* is a configuration where no rule can be applied.

A *computation* is a sequence of transitions between configurations of Π, starting from the initial configuration. A computation is *successful* if and only if it reaches a final configuration or, in other words, it *halts*. It is understood that the multiset (over A, that is, not considering free energy units) of objects which occur in $w_{i_{\text{in}}}$ are the *input values* for the computation. Analogously, the multiset (over A) of objects occurring in the output membrane (or emitted from the skin if $i_{\text{out}} = 0$) in the final configuration is the *output* of the computation. A non–halting computation produces no output.

Since energy is an additive quantity, it is natural to define the *energy of a multiset* as the sum of the amounts of energy associated with each instance of the objects which occur into the multiset. Analogously, the energy of a configuration is the sum of the amounts of energy associated with each multiset which occurs into the configuration. A *conservative computation* is a computation where each configuration has the same amount of energy. A *conservative energy–based P system* is an energy–based P system that performs only conservative computations.

4 Simulating the Fredkin Gate with Energy–based P Systems

In this section we show how P systems, and specifically the energy–based variant introduced in the previous section, can be used to simulate a Fredkin gate.

When trying to simulate a Fredkin gate with a P system, perhaps the simplest idea is to associate a symbol to each possible input/output triple as shown in the table on the left side of Figure 2. Then, the gate is trivially simulated as shown on

Triple	Symbol
$(0,0,0)$	a
$(0,0,1)$	b
$(0,1,0)$	c
$(0,1,1)$	d
$(1,0,0)$	e
$(1,0,1)$	f
$(1,1,0)$	g
$(1,1,1)$	h

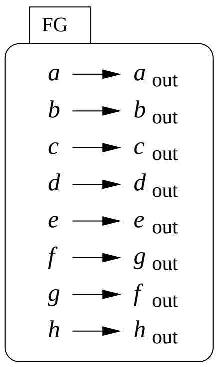

Figure 2: A trivial simulation of the Fredkin gate with a P system. With each possible input/output triple of the gate is associated a symbol of the alphabet

the right side of the same figure: when a symbol corresponding to the input triple is injected into the skin of the P system, in one step the symbol corresponding to the output triple is expelled into the environment. However, this method is not suitable to simulate circuits composed by Fredkin gates. In fact, consider for instance the circuit depicted in Figure 3. The symbol corresponding to the input

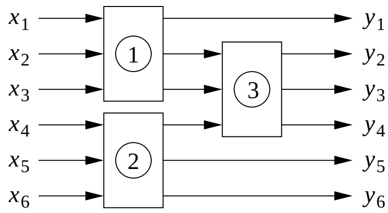

Figure 3: A 6–input/6–output Fredkin circuit composed by three gates

triple of gate number 3 depends upon the symbols corresponding to the output triples of both gates 1 and 2. It is immediately seen that the output symbols of a layer of a Fredkin circuit cannot be immediately used as an input to the next layer: instead, a non trivial transformation is required.

An alternative approach, that solves the previous problem, is to use an energy–based P system as defined in the previous section. The system has 18 objects, with integer energies going from 1 to 18. However, we actually use only 12 objects: precisely, those having energies from 1 to 10 and those having energies 17 and 18. The objects having energies from 11 to 16 never appear into the system. These choices are made in order to have objects with distinct energies and to guarantee conservativeness. For the sake of clarity, we denote the 12 objects used into the system by $[b, j]$ and $[b', j]$, with $b, b' \in \{0, 1\}$ and $j \in \{1, 2, 3\}$. Intuitively, $[b, j]$ and $[b', j]$ indicate the boolean value which occurs in the j-th line of the Fredkin gate. It will be clear from the simulation that we need two different symbols to represent each of these boolean values. The energies are associated with the objects as illustrated in Table 2. In Figure 4 the energy–based P system that simulates the Fredkin gate is depicted.

The simulation works as follows. The input values $[x_1, 1], [x_2, 2], [x_3, 3]$, with $x_1, x_2, x_3 \in \{0, 1\}$, are injected into the skin. If $x_1 = 0$, then the object $[0, 1]$ enters into membrane ID, where it is transformed to the object $[0', 1]$ by releasing 8 units of energy. The object $[0', 1]$ leaves membrane ID and waits for 8 energy units to transform back to $[0, 1]$ and leave the system. The objects $[x_2, 2]$ and $[x_3, 3]$, with $x_2, x_3 \in \{0, 1\}$, may enter nondeterministically either into membrane ID or into membrane EXC; however, if they enter into EXC, then they cannot be transformed to $[x_2', 3]$ and $[x_3', 2]$ since in EXC there are no free energy units. Thus

Object	Energy
$[0,1]$	17
$[1,1]$	18
$[0,2]$	1
$[1,2]$	2
$[0,3]$	3
$[1,3]$	4

Object	Energy
$[0',1]$	9
$[1',1]$	10
$[0',2]$	5
$[1',2]$	6
$[0',3]$	7
$[1',3]$	8

Table 2: Association between objects and energies in the energy–based P system that simulates a Fredkin gate

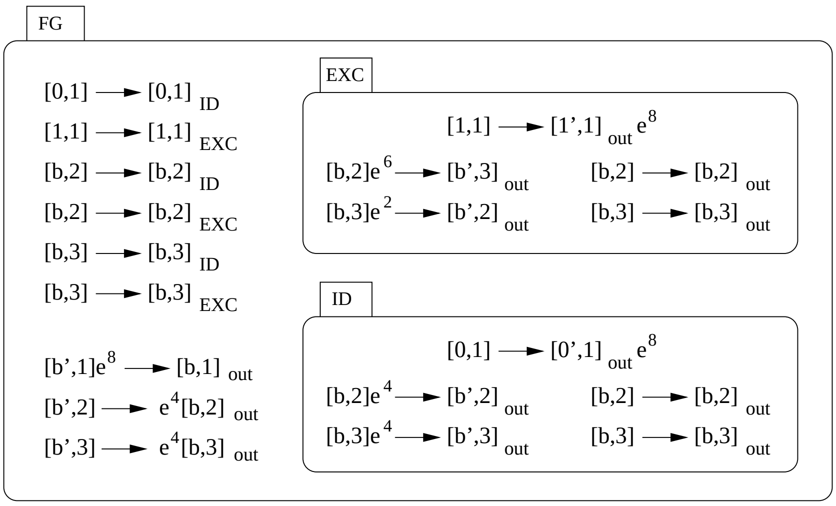

Figure 4: Simulation of the Fredkin gate with an energy–based P system

the only possibility for objects $[x_2, 2]$ and $[x_3, 3]$ is to leave EXC and choose again between membranes ID and EXC in a nondeterministic way. Eventually, after some time they enter (one at the time or simultaneously) into membrane ID. Here they have the possibility to transform to $[x_2', 2]$ and $[x_3', 3]$ respectively, using the 8 units of free energy which occur into the region enclosed by ID (alternatively, they have the possibility to leave ID and choose nondeterministically between membranes ID and EXC once again). When the objects $[x_2', 2]$ and $[x_3', 3]$ are produced they immediately leave ID, and are only allowed to transform back to $[x_2, 2]$ and $[x_3, 3]$ respectively, releasing 8 units of energy. The objects $[x_2, 2]$ and $[x_3, 3]$ just produced leave the system, and the 8 units of energy can only be used to transform $[0', 1]$ back to $[0, 1]$ and expel it from the skin. Hence, as expected, the objects $[x_2, 2]$ and $[x_3, 3]$ initially injected into the skin leave the system unaltered, corresponding to the computation of $\text{FG}(0, x_2, x_3) = (0, x_2, x_3)$.

On the other hand, if $x_1 = 1$ then the object $[1, 1]$ enters into membrane EXC where it is transformed into the object $[1', 1]$ by releasing 8 units of energy. The object $[1', 1]$ leaves the membrane EXC and waits for 8 energy units to transform back to $[1, 1]$ and leave the system. Once again the objects $[x_2, 2]$ and $[x_3, 3]$, with $x_2, x_3 \in \{0, 1\}$, may choose nondeterministically to enter either into membrane ID or into membrane EXC. If they enter into ID they can only exit again since in ID there are no free energy units. When they enter into EXC they can be transformed to $[x_2', 3]$ and $[x_3', 2]$ respectively, using the 8 free energy units which occur into the region, and leave EXC. Now objects $[x_2', 3]$ and $[x_3', 2]$ can only transform to $[x_2, 3]$ and $[x_3, 2]$ respectively, and leave the system. During this transformation 8 free energy units are produced; these can only be used to transform $[1', 1]$ back to $[1, 1]$, which leaves the system. It is immediately seen that the objects $[x_2, 2]$ and $[x_3, 3]$ initially injected into the skin are changed by the system to $[x_2, 3]$ and $[x_3, 2]$, respectively, corresponding to the computation of $\text{FG}(0, x_2, x_3) = (0, x_2, x_3)$.

It is apparent from the simulation that the system can be defined to work on any triple of lines of a circuit, simply modifying the values of the second component of the objects manipulated by the system. Since this component explicitly indicates the position of each boolean value in the input and output triples, it is also easy to rearrange the values produced as the output of a given layer of a circuit in order to produce the input values for the next layer. Such transformations can be made by a P system which uses very simple rules.

Let us note that the proposed P system is conservative: the amount of energy present into the system remains constant during each computation. Precisely, the number of energy units present into the system (both free and embedded into objects) during a computation is 21 plus the number of 1's contained into the input triple of the simulated Fredkin gate. Notice also that the energy of every possible output triple is the same as the energy of the input triple that generated it. The system is also reversible: it is immediately seen that if we inject into the skin the output triple just produced as the result of a computation, the system will expel the corresponding input triple. This behavior is trivially due to the fact that the Fredkin gate is *self–reversible*, meaning that $\text{FG} \circ \text{FG} = \text{ID}_3$ (equivalently, $\text{FG} = \text{FG}^{-1}$), where ID_3 is the identity function on $\{0, 1\}^3$. Notice that, in general, this property does not hold for the functions $f : \{0, 1\}^n \to \{0, 1\}^n$ computed by n–input/n–output Fredkin circuits. Indeed, f is self–reversible if and only if the permutation it applies on the set $\{0, 1\}^n$ can be expressed as a composition of pairwise disjoint transpositions. This means that in general the P system that simulates a given Fredkin circuit must be appropriately designed in order to be itself reversible.

If we drop the assumption that different objects in the system must be associated with different amounts of energy, we can reduce the number of different objects in the system that simulates the Fredkin gate. In the new system, depicted in Figure 5, there are 12 objects. Every object of the kind $[b, j]$, with

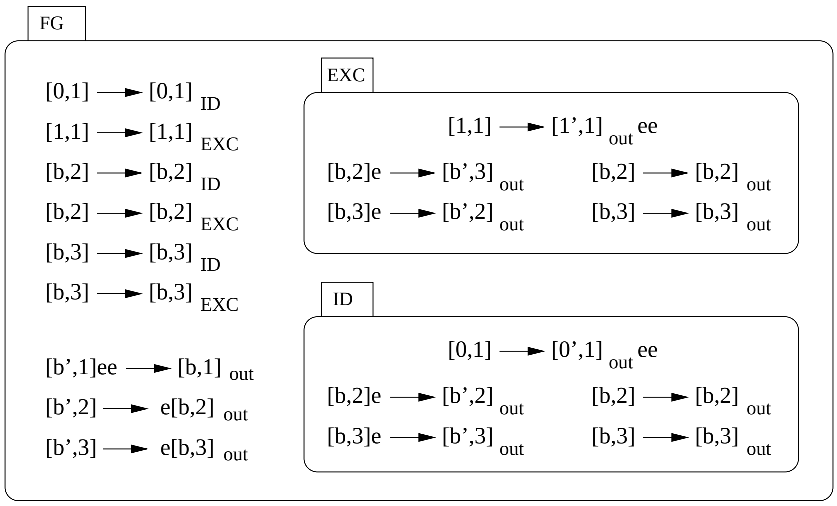

Figure 5: Another simulation of the Fredkin gate with an energy–based P system. Here we drop the assumption that different objects in the system must be associated with different amounts of energy

$b \in \{0, 1\}$ and $j \in \{1, 2, 3\}$, has energy equal to 3, whereas the objects $[b', 1]$ have energy equal to 1 and the objects $[b', 2]$ and $[b', 3]$ (with $b' \in \{0, 1\}$) have energy equal to 4.

Also this system is reversible and conservative. Precisely, when the gate doesn't perform any computation the total amount of energy into the system is zero, whereas during computations the system contains the 9 energy units which have been injected with the input values. At the end of the computation, all these energy units are embedded into the output values. This last simulation is the base upon which we plan to build energy based P systems which simulate Fredkin circuits. This work is still in progress; the results will be published in the near future.

We conclude this section with an observation concerning the use of the special symbol e. In [Ceterchi and Sburlan 2003] it is shown how P systems can be used to simulate boolean circuits composed by the gates AND, OR and NOT. The simplest simulation uses context–sensitive (cooperative) rules. In order to avoid context–sensitive rules mobile catalysts are introduced, with or without the use of promoters and of weak priorities between rules. Here we note that the use of free energy units, as we have done to simulate the Fredkin gate, seems to be equivalent to the assumption that e is the only symbol that may cooperate with all the other symbols. In other words, we do not allow generic cooperative rules but only those with the special symbol $e \notin A$ cooperating with any symbol of A. On the other hand, it seems that cooperation is necessary to simulate the Fredkin gate, since the value of the second and third output lines of the gate depend upon all three input values. Cooperation is also necessary for the AND and OR gates, although the fact that their output value only depends upon the number of 1's given in input would seem an evidence of the contrary. For if this were not true, then we could easily build a Fredkin gate using AND, OR and NOT gates, according to equations (1), and realize a cooperative behavior without using cooperation.

Let us note also that in case of necessity (for example, during proofs) we can safely assume that for each rule at most one instance of e cooperates with a symbol of the alphabet. In fact, any rule of the kind $a \to (b, p)e^k$, with $a, b \in A$ and $p \in \{\text{here}, \text{in}(name), \text{out}\}$, involving k instances of e, can be decomposed as follows:

$$\begin{aligned}
&a \to (b_1, \text{here})e, \\
&b_1 \to (b_2, \text{here})e, \\
&\quad\vdots \\
&b_{k-2} \to (b_{k-1}, \text{here})e, \\
&b_{k-1} \to (b, p)e,
\end{aligned}$$

by introducing into the alphabet the *new* symbols $b_1, \ldots, b_{k-1}$. An analogous observation holds for rules of the kind $ae^k \to b$.

5 Some Possible Extensions of the Model, and Corresponding Open Problems

The model of energy–based P system introduced in section 3 can be extended in many ways. All the features mentioned below can be introduced independently of each other. Of course, for each possible extension we advocate the study of the computational capabilities of the resulting model of computation, as well as of the algebraic and language–theoretic properties of the generated multiset languages.

As a first extension, we can assume that every membrane and every rule possesses a given amount of energy. The only use of membrane energy we are able to imagine is that membranes may act as energy reservoirs. This means that membranes can incorporate free energy units in their internal structure and subsequently release them. We can also assume that a membrane must possess a positive amount of energy in order to exist; in other words, we can assume that when the energy of a membrane becomes zero the membrane dissolves. In the case an output membrane is used (that is, when $i_{\text{out}} \neq 0$) it seems reasonable to assume that such a membrane cannot dissolve, that is, that it cannot release all its internal energy. Moreover, an interesting constraint could be putting a fixed upper bound on the energy that a membrane can embed. A further constraint could be to divide objects and membranes into *types*, for example, based upon morphological characteristics, and to allow the system to exchange energy only between objects (and/or membranes) of the same type. As for the energy associated with rules, we can assume that every rule is defined together with a fixed threshold value. If the energy of the rule does not reach this value, then the rule is *inactive*. This means that even if all the objects mentioned in the left side of the rule are available, the rule cannot be applied until enough free energy units become available to allow the threshold to be reached.

Two open problems concerning energy associated with membranes and/or to rules are the following: are these models of energy–based P systems able to simulate in an efficient way P systems which use priorities in their rules? Is it possible to give an efficient simulation of P systems whose membranes have an associated thickness or polarization, as the ones defined in [Zandron et al. 2000, Zandron et al. 2001]?

Another possible extension of the model could be allowing the use of constructor and destructor rules. A *constructor rule* is a rule of the kind $e^k \to (a, p)$, where $a \in A$, $\varepsilon_a = k$, $p \in \{\text{here}, \text{in}(name), \text{out}\}$, and k is a positive integer. Informally, a constructor rule for an object $a \in A$ is a rule which uses ε_a free energy units to build the object a. In other words, we allow transformations from "pure" energy to system objects. Analogously, a *destructor rule* is a rule of the kind $a \to e^k$, where $a \in A$ and $\varepsilon_a = k$ (a positive integer). Hence, a destructor rule for an object $a \in A$ is a rule which transforms the object a into ε_a units of free energy.

We can define the total energy of an energy–based P system as follows. As in the basic version, the energy of a multiset is simply the sum of the amounts of energy associated with each instance of the objects which occur into the multiset. The total energy of a membrane is the sum of the energies associated with the multisets contained into the region enclosed by the membrane, plus the energies associated with the rules of the region, plus the amount of energy embedded into the membrane itself. It is understood that the total energy of a membrane i comprises the sum of total energies of the membranes contained in i in the hierarchy μ. The total energy of a P system can thus be defined as the total energy of the skin.

Where does the energy come from? We assume that there is an external (with respect to the skin) *reservoir* of energy that injects free energy units into the skin at a constant rate. The computation halts when there is no rule which can be applied, even in presence of additional free energy. A natural question is whether this last additional constraint is necessary. In other words, are systems defined with and without this additional condition equivalent with respect to generated multiset languages? Stated otherwise: for any given system which does not satisfy this condition, can we build a system which satisfies it and yet generates the same multiset language? Notice that if the above halting condition should prove too bothersome to deal with, we can alternatively assume that a computation halts when a given condition is verified, for example, when an object enters an *acknowledgment* membrane, as it happens in conformon–P systems [Frisco 2004].

An interesting situation arises when we associate a *clock* to the external energy reservoir, so that one energy unit is injected into the system at each clock tick. In this situation objects and membranes are usually *quiescent*, that is they do not evolve, because they do not have enough energy to do it. The unit of energy coming from the reservoir allows to apply at most one evolution rule at the time. This fact leads us to define "sequential" P systems, as opposed to the usual parallel models. An interesting issue could be the comparison between the computational properties of sequential and parallel P systems. Can they always be simulated by each other?

A further aspect of controlling the amount of energy injected into the system by the external reservoir is the following. By assuming that at each clock tick a fixed number of energy units is injected into the system, we are somehow imposing a constraint on the level of parallelism of the system. However, this assumption should be considered with care since the application of rules may free some energy units that move into the system and temporarily rise the level of parallelism. As a consequence, the computational power of this kind of systems could become really hard to explore.

Due to the number of constraints and parameters we have introduced into our model, it could be difficult to study their computational behavior and/or deduce interesting properties. Hence we propose to start to study energy–based P systems basing upon a simpler model, namely P systems with symport/antiport rules. In these systems the objects cannot be modified but only moved between regions. Moreover, we can first assume that the application of rules does not consume energy, and that there is no energy embedded into the internal structure of membranes. Subsequently, we could study how the introduction of these features alter the computational behavior of the corresponding systems.

As for the computational power of energy–based P systems we propose the introduction of languages which can be generated using a bounded (fixed, logarithmic, polynomial, etc.) amount of energy, and the subsequent investigation of the properties of these languages. A second proposal concerns the introduction of conservative energy–based P systems and the study of their computational properties. A natural measure of the energy used by a system is the amount of energy which is injected into the system by the external reservoir. Notice that in the case of P systems with symport/antiport rules with just one symbol into the alphabet, this complexity measure is the same as the number of objects which enter into the system. The measure becomes less trivial even for P systems with symport/antiport rules having at least two kinds of objects. Eventually, some free energy units remain into the system at the end of the computation: can this situation be avoided? It is not clear to the authors whether this amount of energy should be considered as "consumed" energy or whether it can be expelled from the skin and thus "recovered". In the latter situation, we could define *conservative* energy–based P systems as systems that, at the end of the computation, have emitted the same quantity of energy that entered the system during the computation.

A different approach to study the computational power of energy–based P systems is to define *families* $\{P_n\}_{n\in\mathbb{N}}$ of energy–based P systems, where P_n uses n units of energy. Then, we can define the language generated by $\{P_n\}_{n\in\mathbb{N}}$ as $\bigcup_{n\in\mathbb{N}} L_n$, where L_n is the language generated by P_n. This approach is reminiscent of circuit complexity [Vollmer 1999]. Moreover, having defined both an input and an output membrane, we can view energy–based P systems as devices which map multisets into multisets. With respect to this point of view, instead of asking what multisets can be generated by the system we can ask what mappings can be realized by imposing different bounds on the amount of resources that the system is allowed to use.

A powerful approach to study the properties of any computational model is to look at the amount of resources which are needed to simulate it using another model of computation. In particular, simulations with counter machines have proven to be a powerful tool to study P systems. We think that counter machines should be the reference model also for energy–based P systems. In particular, it seems natural to think that the energy levels of objects, rules and membranes can be simulated by values contained into counters. Does this mean that energy bounded computations correspond to computations performed by counter machines with bounds on the values of their counters?

Finally, we propose to compare the computational properties of our model with those of conformon–P systems [Frisco 2004] (see also [Frisco and Ji 2002a] and [Frisco and Ji 2002b]). Notice that our approach in the use of energy is different from the one followed in [Frisco 2004]: in that paper, energy is a feature of objects (named *conformons*) whereas in our model the amount of energy embedded into an object *determines* the type of the object, since there is a bijective correspondence between objects and the amounts of energy they embed.

6 Conclusions and Directions for Future Work

In this paper we have defined a basic version of energy–based P systems, that is, P systems in which the amount of energy manipulated during computations is taken into account. We have also defined the notion of conservative energy–based P system.

Two simulations of the Fredkin gate have been presented using this new model of computation. The P systems that perform the simulations turn out to be themselves reversible and conservative as the Fredkin gate. Subsequently we have proposed some possible extensions to the model, with a number of open problems concerning their computational power.

The next logical step in our work is to simulate reversible Fredkin circuits using our basic version of energy–based P systems. Our guess is that the simulating P system can be made both reversible and conservative. Here reversible means that, if the output values of the Fredkin circuit are given, the same system is able to compute the corresponding input values.

Acknowledgements

The present paper has been inspired by [Ceterchi and Sburlan 2003] and by a question posed by Fernando Sancho Caparrini during the Second Brainstorming Week on Membrane Computing held in Seville from 2 to 7 of February 2004.

We are also indebted to Pierluigi Frisco for stimulating discussion about energy–based P systems, conformons and reversibility.

This work was partially supported by contribution of EU Commission under the Fifth Framework Programme, project "MolCoNet" IST-2001-32008.

References

[Alford 2002] Alford G.: "Membrane Systems with Heat Control"; "Pre–Proceedings of Workshop on Membrane Computing", Curtea de Arges, Romania, August 2002, MolCoNet Publication No 1. (2002) `http://psystems.disco.unimib.it/VolumeContents/WMC02.pdf`

[Bekenstein 1981] Bekenstein J.D.: "Energy Cost of Information Transfer"; Physical Review Letters, 46, 10 (1981), 623–626

[Benioff 1982] Benioff P.: "Quantum Mechanical Models of Turing Machines that Dissipate no Energy"; Physical Review Letters, 48, 23 (1982) 1581–1585

[Bennett 1973] Bennett C.H.: "Logical Reversibility of Computation"; IBM Journal of Research and Development, 17 (Nov. 1973) 525–532

[Cattaneo et al. 2002a] Cattaneo G., Della Vedova G. Leporati A., Leporini R.: "Towards a Theory of Conservative Computing"; International Journal of Theoretical Physics, to appear. Preprint available at `http://arxiv.org/abs/quant-ph/0211085`, (Nov. 2002)

[Cattaneo et al. 2002b] Cattaneo G., Leporati A., Leporini R.: "Fredkin Gates for Finite–valued Reversible and Conservative Logics"; Journal of Physics A: Mathematical and General, 35 (Nov. 2002), 9755–9785

[Cattaneo et al. 2001] Cattaneo G., Leporati A., Leporini R.: "Quantum Conservative Gates for Finite–valued Logics"; International Journal of Theoretical Physics, to appear

[Ceterchi and Sburlan 2003] Ceterchi R., Sburlan D.: "Simulating Boolean Circuits with P Systems"; "Membrane Computing, Proceedings of the International Workshop WMC 2003", Tarragona, Spain, July 2003, Lecture Notes in Computer Science 2933, Springer (2003), 104–122

[Erk 1999] Erk K.: "Simulating Boolean Circuits by Finite Splicing"; Proceedings of the Congress on Evolutionary Computation, IEEE Press, 2, 6-9 (1999), 1279–1285

[Fredkin and Toffoli 1982] Fredkin E., Toffoli T.: "Conservative Logic"; International Journal of Theoretical Physics, 21, 3-4 (1982), 219–253

[Freund 2002] Freund R.: "Energy–Controlled P Systems"; "Membrane Computing, Proceedings of the International Workshop WMC–CdeA 2002", Curtea de Arges, Romania, August 2002, Lecture Notes in Computer Science 2597, Springer (2002), 247–260

[Freund et al. 2004] Freund R., Leporati A., Oswald M., Zandron C.: "Sequential P Systems with Unit Rules and Energy Assigned to Membranes"; "Second Week on Membrane Computing"; Sevilla, Spain, Feb. 2–7, 2004, Research Group on Natural Computing, University of Sevilla, Technical Report 01/2004, (2004), 168–182

[Frisco 2004] Frisco P.: "The Conformon–P System: A Molecular and Cell Biology–Inspired Computability Model"; Theoretical Computer Science, 312 (2004), 295–319

[Frisco and Ji 2002a] Frisco P., Ji S.: "Conformons–P Systems"; "DNA Computing, Proceedings of 8th International Workshop on DNA–Based Computers, DNA 8", Sapporo, Japan, June 2002, Lecture Notes in Computer Science 2568, Springer (2002), 291–301

[Frisco and Ji 2002b] Frisco P., Ji S.: "Towards a Hierarchy of Conformons–P Systems"; "Membrane Computing, Proceedings of the International Workshop WMC–CdeA 2002", Curtea de Arges, Romania, August 2002, Lecture Notes in Computer Science 2597, Springer (2002), 302–318

[Ji 2001] Ji S.: "The Bhopalator: An Information/Energy Dual Model of the Living Cell"; Fundamenta Informaticae, 49, 1-3 (2002), 147–165

[Klein et al. 1999] Klein J.P., Leete T.H., Rubin H.: "A Biomolecular Implementation of Logically Reversible Computation with Minimal Energy Dissipation"; BioSystems, 52 (1999) 15–23

[Landauer 1961] Landauer R.: "Irreversibility and Heat Generation in the Computing Process"; IBM Journal of Research and Development, 5 (1961) 183–191

[Landauer 1982] Landauer R.: "Uncertainty Principle and Minimal Energy Dissipation in the Computer"; International Journal of Theoretical Physics, 21, 3-4 (1982) 283–297

[Mauri and Leporati 2003] Mauri G., Leporati A.: "On the Computational Complexity of Conservative Computing"; "Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science, MFCS 2003", Lecture Notes in Computer Science 2747, Springer (2003), 92–112

[Ogihara and Ray 1996] Ogihara M., Ray A.: "Simulating Boolean Circuits on a DNA Computer"; Technical Report 631 (1996). `http://citeseer.nj.nec.com/ogihara96simulating.html`

[Păun 1998] Păun Gh.: "Computing with Membranes"; Journal of Computer and System Sciences, 1, 61 (2000) 108–143

[Păun 1999] Păun Gh.: "Computing with Membranes. An Introduction"; Bulletin of the EATCS, 67 (Feb. 1999) 139–152

[Păun 2000] Păun Gh.: "Computing with Membranes. A Variant: P Systems with Polarized Membranes"; International Journal on Foundations of Computer Science, 11, 1 (2000), 167–182 `http://www.cs.auckland.ac.nz/CDMTCS`

[Păun 2002] Păun Gh.: "Membrane Computing. An Introduction"; Springer, Berlin (2002)

[Păun and Rozenberg 2002] Păun Gh., Rozenberg Gh.: "A Guide to Membrane Computing"; Theoretical Computer Science, 287, 1 (2002), 73–100

[Păun et al. 2001] Păun Gh., Suzuki Y., Tanaka H.: "P Systems with Energy Accounting"; International Journal Computer Math., 78, 3 (2001), 343–364

[Petri 1965] Petri C.A.: "Gründsatzliches zur Beschreibung diskreter Prozesse"; Proceedings of the 3rd Colloquium über Automatentheorie, Hannover (1965), Birkhäuser Verlag, Basel (1967), 121–140. English translation: Fundamentals of the Representation of Discrete Processes, ISF Report 82.04 (1982)

[Prakash and Krithivasan 2004] Prakash V.J., Krithivasan K.: "Simulating Boolean Circuits with Tissue P Systems"; manuscript (2004)

[P systems Web page] The P systems Web page: `http://psystems.disco.unimib.it/`

[Vollmer 1999] Vollmer H.: "Introduction to Circuit Complexity: A Uniform Approach"; Springer, Berlin (1999)

[Zandron et al. 2001] Zandron C., Ferretti C. Mauri, G.: "Using Membrane Features in P Systems"; Romanian Journal of Information Science and Technology, 4, 1-2 (2001), 241–257

[Zandron et al. 2000] Zandron C., Mauri G., Ferretti C.: "Universality and Normal Forms on Membrane Systems" (R. Freund, A. Kelemenova, eds.), "Proceedings of the International Workshop on Grammar Systems", July 2000, Bad Ischl, Austria (2000), 61–74