Solving PP-Complete and #P-Complete Problems by P Systems with Active Membranes

Artiom Alhazov[1], Liudmila Burtseva[1], Svetlana Cojocaru[1], and Yurii Rogozhin[1,2]

[1] Academy of Sciences of Moldova
Institute of Mathematics and Computer Science
Academiei 5, MD-2028, Chişinău, Moldova
{artiom,burtseva,sveta,rogozhin}@math.md
[2] Rovira i Virgili University
Research Group on Mathematical Linguistics
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain

Abstract. Membrane computing is a formal framework of distributed parallel multiset processing. Due to massive parallelism and exponential space some intractable computational problems can be solved by P systems with active membranes in a polynomial number of steps. In this paper we generalize this approach from decisional problems to the computational ones, by providing a solution of a **#P**-complete problem, namely to compute the permanent of a binary matrix. The implication of this result to the **PP** complexity class is discussed and compared to known results about $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP}$.

1 Introduction

Membrane systems are a convenient framework of describing polynomial-time solutions to certain intractable problems in a massively parallel way. Division of membranes makes it possible to create an exponential space in linear time, suitable for attacking problems in **NP** and even in **PSPACE**. Their solutions by so-called P systems with active membranes have been investigated in a number of papers since 2001, later focusing on solutions by restricted systems.

The description of rules in P systems with active membranes involves membranes and objects; the typical types of rules are (a) object evolution, $(b), (c)$ object communication, (d) membrane dissolution, (e) membrane division – see Subsection 2.2. Since membrane systems are an abstraction of living cells, the membranes are arranged hierarchically, yielding a tree structure. A membrane is called elementary if it is a leaf of this tree, i.e., if it does not contain other membranes.

The first efficient *semi–uniform solution* to SAT was given in [4], using division for non–elementary membranes and three electrical charges. This result was improved in [5] using only division for elementary membranes.

Different efficient *uniform solutions* have been obtained in the framework of recognizer P systems with active membranes, with polarizations and only using

D. Corne et al. (Eds.): WMC9 2008, LNCS 5391, pp. 108–117, 2009.

division rules for elementary membranes (see, e.g., [8], [6], [3], [7], [9] and their references).

The goal of this paper is to generalize the approach from decisional problems to the computational ones, by considering a **#P**-complete (pronounced sharp-P complete) problem of computing the *permanent* of a binary matrix; see also Section 1.3.7 in [11] for a presentation of Complexity Theory of counting problems.

Let us cite [12] for additional motivation:

> While 3SAT and the other problems in **NP**-complete are widely assumed to require an effort at least proportional to 2^n, where n is a measure of the size of the input, the problems in **#P**-complete are harder, being widely assumed to require an effort proportional to $n2^n$.

While attacking **NP** complexity class by P systems with active membranes have been often motivated by $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ problem, we recall from [13] the following fact:

> If the permanent can be computed in polynomial time by any method, then **FP=#P** which is an even stronger statement than **P= NP**.

Here, by "any method" one understands "... on sequential computers" and **FP** is the set of polynomial-computable functions.

In Section 4 we recall the definition of **PP** (the probabilistic polynomial time complexity class) and present an approach to solving the problems in **PP**.

2 Definitions

Membrane computing is a recent domain of natural computing started by Gh. Păun in 1998. The components of a membrane system are a cell-like membrane structure, in the regions of which one places multisets of objects which evolve in a synchronous maximally parallel manner according to given evolution rules associated with the membranes.

2.1 Computing by P Systems

Let O be a finite set of elements called objects. In this paper, like it is standard in membrane systems literature, a multiset of objects is denoted by a string, so the multiplicity of object is represented by number of its occurrences in the string. The empty multiset is thus denoted by the empty string, λ.

To speak about the result of the computation of a P system we need the definition of a P system with output.

Definition 1. *A P system with output, Π, is a tuple*

$\Pi = \big(O, T, H, E, \mu, w_1, \cdots, w_p, R, i_0\big)$, *where:*

- *O is the working alphabet of the system whose elements are called objects.*
- *$T \subseteq O$ is the output alphabet.*
- *H is an alphabet whose elements are called labels.*

- E *is the set of polarizations.*
- μ *is a membrane structure (a rooted tree) consisting of* p *membranes injectively labeled by elements of* H*.*
- w_i *is a string representing an* initial *multiset over* O *associated with membrane* i*,* $1 \leq i \leq p$*.*
- R *is a finite set of rules defining the behavior of objects from* O *and membranes labeled by elements of* H*.*
- i_0 *identifies the output region.*

A configuration of a P system is its "snapshot", i.e., the current membrane structure and the multisets of objects present in regions of the system. While initial configuration is $C_0 = (\mu, w_1, \cdots, w_p)$, each subsequent configuration C' is obtained from the previous configuration C by maximally parallel application of rules to objects and membranes, denoted by $C \Rightarrow C'$ (no further rules are applicable together with the rules that transform C into C'). A computation is thus a sequence of configurations starting from C_0, respecting relation $\Rightarrow$ and ending in a halting configuration (i.e., one where no rules are applicable).

The P systems of interest here are those for which all computations give the same result. This is because it is enough to consider one computation to obtain all information about the result.

Definition 2. *A P system with output is* confluent *if (a) all computations halt; and (b) at the end of all computations of the system, region* i_0 *contains the same multiset of objects from* T*.*

In this case one can say that the multiset mentioned in (b) is the result given by a P system, so this property is already sufficient for a convenient usage of P systems for computation.

However, one can still speak about a stronger property: a P system is *strongly confluent* if not only the result of all computation is the same, but also the halting configuration is the same. A yet stronger property is determinism: a P system is called *deterministic* if it only has one computation.

In what follows we represent computational problems by triples: domain, range and the function (from that domain into that range) that needs to be computed. The notation $\mathbf{PMC}^*_{\mathcal{R}}$ of the class of problems that are polynomially computable by semi-uniform families of P systems with active membranes has been introduced by M.J. Pérez-Jiménez and his group, see, e.g., [8]. The definition below generalizes it from decisional problems to computational ones.

Definition 3. *Let* $X = (I_X, F, \theta_X)$ *be a computational problem:* $\theta_X : I_X \to F$*. We say that* X *is solvable in polynomial time by a (countable) family* $\mathcal{R}$ *of confluent P systems with output* $\mathbf{\Pi} = (\Pi(u))_{u \in I_X}$*, and we denote this by* $X \in \mathbf{PMC}^*_{\mathcal{R}}$*, if the following are true.*

1 *The family* $\mathbf{\Pi}$ *is polynomially uniform by Turing machines, i.e., there exists a deterministic Turing machine working in polynomial time which constructs the system* $\Pi(u)$ *from the instance* $u \in I_X$*.*

2 The family $\mathbf{\Pi}$ *is polynomially bounded: for some polynomial function* $p(n)$ *for each instance* $u \in I_X$ *of the problem, all computations of* $\Pi(u)$ *halt in, at most,* $p(|u|)$ *steps.*

3 There exists a polynomial-time computable function dec such that the family $\mathbf{\Pi}$ *correctly answers* X *with respect to* (X, dec)*: for each instance of the problem* $u \in I_X$*, the function dec applied to the result given by* $\Pi(u)$ *returns exactly* $\theta_X(u)$*.*

We say that the family $\mathbf{\Pi}$ is a *semi–uniform solution* to the problem X.

Now we additionally consider input into P systems and we deal with P systems solving computational problems in a *uniform* way in the following sense: all instances of the problem with the same *size* (according to a previously fixed polynomial time computable criterion) are processed by the same system, on which an appropriate input, representing the specific instance, is supplied.

If w is a multiset over the input alphabet $\Sigma \subseteq O$, then the *initial configuration* of a P system Π with an input w over alphabet Σ and input region i_Π is

$$(\mu, w_1, \cdots, w_{i_\Pi - 1}, w_{i_\Pi} \cup w, w_{i_\Pi + 1} \cdots, w_p).$$

In the definition below we present the notation $\mathbf{PMC}_{\mathcal{R}}$ of the class of problems that are polynomially computable by uniform families of P systems with active membranes introduced by M.J. Pérez-Jiménez and his group, see, e.g., [8], generalized from decisional problems to computational ones.

Definition 4. *Let* $X = (I_X, F, \theta_X)$ *be a computational problem. We say that* X *is solvable in polynomial time by a family* $\mathbf{\Pi} = (\Pi(n))_{n \in \mathbb{N}}$ *of confluent membrane systems with input, and we denote it by* $X \in \mathbf{PMC}_{\mathcal{R}}$*, if*

1 The family $\mathbf{\Pi}$ *is polynomially uniform by TM: some deterministic TM constructs in polynomial time the system* $\Pi(n)$ *from* $n \in \mathbb{N}$*.*

2 There exists a pair (cod, s) *of polynomial-time computable functions whose domain is* I_X *and a polynomial-time computable function dec whose range is* F*, such that for each* $u \in I_X$*,* $s(u)$ *is a natural number,* $cod(u)$ *is an input multiset of the system* $\Pi(s(u))$*, verifying the following:*

2a The family $\mathbf{\Pi}$ *is polynomially bounded with respect to* (X, cod, s)*; that is, there exists a polynomial function* $p(n)$ *such that for each* $u \in I_X$ *every computation of the system* $\Pi(s(u))$ *with input* $cod(u)$ *halts in at most* $p(|u|)$ *steps.*

2b There exists a polynomial-time computable function dec such that the family $\mathbf{\Pi}$ *correctly answers* X *with respect to* (X, cod, s, dec)*: for each instance of the problem* $u \in I_X$*, the function dec, being applied to the result given by* $\Pi(s(u))$ *with input* $cod(u)$*, returns exactly* $\theta_X(u)$*.*

We say that the family $\mathbf{\Pi}$ is a *uniform solution* to the problem X.

2.2 P Systems with Active Membranes

To speak about P systems with active membranes, we need to specify the rules, i.e., the elements of the set R in the description of a P system. They can be of the following forms:

(a) $[\ a \to v\ ]_h^e$, for $h \in H, e \in E, a \in O, v \in O^*$
(object evolution rules, associated with membranes and depending on the label and the polarization of the membranes, but not directly involving the membranes, in the sense that the membranes are neither taking part in the application of these rules nor are they modified by them);

(b) $a[\ \ ]_h^{e_1} \to [\ b\ ]_h^{e_2}$, for $h \in H, e_1, e_2 \in E, a, b \in O$
(communication rules; an object is introduced into the membrane; the object can be modified during this process, as well as the polarization of the membrane can be modified, but not its label);

(c) $[\ a\ \ ]_h^{e_1} \to [\ \ ]_h^{e_2} b$, for $h \in H, e_1, e_2 \in E, a, b \in O$
(communication rules; an object is sent out of the membrane; the object can be modified during this process; also the polarization of the membrane can be modified, but not its label);

(d) $[\ a\ \ ]_h^e \to b$, for $h \in H, e \in E, a, b \in O$
(dissolving rules; in reaction with an object, a membrane can be dissolved, while the object specified in the rule can be modified);

(e) $[\ a\ \ ]_h^{e_1} \to [\ b\ \ ]_h^{e_2} [\ c\ \ ]_h^{e_3}$, for $h \in H, e_1, e_2, e_3 \in E, a, b, c \in O$
(division rules for elementary membranes; in reaction with an object, the membrane is divided into two membranes with the same label, possibly of different polarizations; the object specified in the rule is replaced in the two new membranes by possibly new objects).

In this paper we do not need division, dissolution or rules that bring an object inside a membrane, but they are mentioned in the definition for completeness.

The rules of type (a) are considered to only involve objects, while all other rules are assumed to involve objects and membranes mentioned in their left-hand side. An application of a rule consists in subtracting a multiset described in the left-hand side from a corresponding region (i.e., associated to a membrane with label h and polarization e for rules of types (a) and (d), or associated to a membrane with label h and polarization e_1 for rules of type (c) and (e), or immediately outer of such a membrane for rules of type (b)), adding a multiset described in the right-hand side of the rule to the corresponding region (that can be the same as the region from where the left-hand side multiset was subtracted, immediately inner or immediately outer, depending on the rule type), and updating the membrane structure accordingly if needed (changing membrane polarization, dividing or dissolving a membrane).

The rules can only be applied simultaneously if they involve different objects and membranes (we repeat that rules of type (a) are not considered to involve a membrane), and such parallelism is maximal if no further rules are applicable to objects and membranes that were not involved.

2.3 Permanent of a Matrix

The complexity class **#P**, see [15], was first defined in [10] in a paper on the computation of the permanent.

Definition 5. *Let S_n be the set of permutations of integers from 1 to n, i.e., the set of bijective functions $\sigma : \{1, \cdots, n\} \to \{1, \cdots, n\}$. The permanent of a matrix $A = (a_{i,j})_{1 \le i,j \le n}$ is defined as*

$$perm(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

Informally, consider a combination of n matrix elements containing one element from every row and one element from every column. The permanent is the sum over all such combinations of the product of the combination's elements.

A matrix is binary if its elements are either 0 or 1. In this case, the permanent is the number of combinations of n matrix elements with value 1, containing one element from each row and one element from each column. For example,

$$perm \begin{pmatrix} 1\,0\,1 \\ 0\,1\,0 \\ 1\,0\,1 \end{pmatrix} = 2.$$

Unlike the determinant of a matrix, the permanent cannot be computed by Gauss elimination.

3 Main Result

Theorem 1. *The problem of computing the permanent of a binary matrix is solvable in polynomial time by a uniform family of deterministic P systems with active membranes with two polarizations and rules of types $(a), (c), (e)$.*

Proof. Let $A = (a_{i,j})$ be an $n \times n$ matrix. We define $N = \lceil \log_2(n) \rceil$, and $n' = 2^N < 2n$ is the least power of two not smaller then n. The input alphabet is $\Sigma(n) = \{\langle i, j \rangle \mid 1 \le i \le n,\ 1 \le j \le n\}$, and the matrix A is given as a multiset $w(A)$ containing for every element $a_{i,j} = 1$ of the matrix one symbol $\langle i, j \rangle$. Let the output alphabet be $T = \{o\}$, we will present a P system $\Pi(n)$ giving $o^{perm(A)}$ as the result when given input $w(A)$ in region $i_{\Pi(n)} = 2$.

$$\begin{aligned}
\Pi(n) &= \big(O, T, H, E, \mu, w_1, w_2, R, 1\big), \\
O &= \Sigma(n) \cup T \cup \{c\} \cup \{d_i, a_i \mid 0 \le i \le Nn\} \cup \{D_i \mid 0 \le i \le n+1\} \\
&\quad \cup \{\langle i, j, k, l \rangle \mid 0 \le i \le Nn-1,\ 0 \le j \le n-1,\ 0 \le k \le Nn-1, \\
&\qquad 0 \le l \le n'-1\}, \\
\mu &= [\,[\,\,]_2^0\,]_1^0,\ H = \{1, 2\},\ E = \{0, 1\}, \\
w_1 &= \lambda,\ w_2 = d_0.
\end{aligned}$$

and the rules are presented and explained below.

A1 $[\ \langle i,j\rangle \to \langle Ni-1, j-1, Nn-1, 0\rangle\ ]_2^0$, $1 \le i \le n$, $1 \le j \le n$

Preparation of the input objects: tuple representation. Informal meaning of the tuple components is 1) number of steps remaining until row i is processed, 2) column number, starting from 0, 3) number of steps remaining until all rows are processed, 4) will be used for memorizing the chosen column.

A2 $[\ d_i\ ]_2^e \to [\ d_{i+1}\ ]_2^0 [\ d_{i+1}\ ]_2^1$, $0 \le i \le Nn-1$, $e \in E$

Division of the elementary membrane for Nn times.

A3 $[\ \langle i,j,k,l\rangle \to \langle i-1, j, k-1, 2l+e\rangle\ ]_2^e$,
$0 \le i \le Nn-1$, i is not divisible by N,
$0 \le j \le n-1$, $1 \le k \le Nn-1$, $0 \le l \le (n-1-e)/2$, $e \in E$

For i times, during $N-1$ steps input objects corresponding to row i memorize the polarization history. The binary representation of the chosen column for the current row corresponds to the history of membrane polarizations during N steps.

A4 $[\ \langle i,j,k,l\rangle \to \lambda\ ]_2^e$,
$0 \le i \le Nn-1$, $0 \le j \le n-1$, $1 \le k \le Nn-1$,
$(n-1-e)/2 \le l \le n'/2-1$, $e \in E$

Erase all input objects if the chosen column is invalid, i.e., its number exceeds $n-1$.

A5 $[\ \langle i,j,k,l\rangle \to \langle i-1, j, k-1, 0\rangle\ ]_2^e$,
$1 \le i \le Nn-1$, $0 \le j \le n-1$, $j \ne 2l+e$,
$0 \le k \le Nn-1$, $0 \le l \le (n-1-e)/2$, $e \in E$

If element's row is not reached and element's column is not chosen, proceed to the next row.

A6 $[\ \langle i,j,k,l\rangle \to \lambda\ ]_2^e$,
$1 \le i \le Nn-1$, $0 \le j \le n-1$, $j = 2l+e$,
$0 \le k \le Nn-1$, $0 \le l \le (n-1-e)/2$, $e \in E$

Erase the chosen column, except the chosen element.

A7 $[\ \langle 0,j,k,l\rangle \to \lambda\ ]_2^e$,
$0 \le j \le n-1$, $j \ne 2l+e$,
$0 \le k \le Nn-1$, $0 \le l \le (n-1-e)/2$, $e \in E$

Erase the chosen row, except the chosen element.

A8 $[\ \langle 0,j,k,l\rangle \to a_{k-1}\ ]_2^e$,
$0 \le j \le n-1$, $j = 2l+e$,
$0 \le k \le Nn-1$, $0 \le l \le (n-1-e)/2$, $e \in E$

If chosen element is present (i.e., it has value 1 and its column has not been chosen before), produce object a_{k-1}.

A9 $[\ a_k \to a_{k-1}\ ]_2^e$, $1 \le k \le Nn-1$, $e \in E$

Objects a_k wait until all rows are processed. Then a membrane represents a solution if n copies of a_0 are present.

B1 $[\ d_{Nn} \to D_{1-e}c^{n+e}\ ]_2^e$, $e \in E$

If polarization is 0, produce n copies of object c and a counter D_1. Otherwise, produce one extra copy of c and set the counter to D_0; this will reduce to the previous case in one extra step.

B2 $[\ c\ ]_2^1 \to [\ \ ]_2^0 c$

B3 $[\ a_0\ ]_2^0 \to [\ \ ]_2^1 a_0$

B4 $[\ D_i \to D_{i+1}\ ]_2^1$, $0 \le i \le n$

Each object a_0 changes polarization to 1, the counter D_i counts this, and then object c resets the polarization to 0.

B5 $[\ D_{n+1}\ ]_2^1 \to [\ \ ]_2^0 o$

If there are n chosen elements with value 1, send one object o out.

The system is deterministic. Indeed, for any polarization and any object (other than d_i, $i < Nn$, c, a_0 or D_{n+1}), there exist at most one rule of type (a) and no other associated rules. As for the objects in parentheses above, they have no rules of type (a) associated with them and they cause a well-observed deterministic behavior of the system: division rules are applied during the first Nn steps; then, depending on the polarization, symbols a_0 or c are sent out; finally, wherever D_{n+1} is produced, it is sent out.

The system computes the permanent of a matrix in at most $n(2+N)+1 = O(n \log n)$ steps. Indeed, the first Nn steps correspond to membrane divisions corresponding to finding all permutations of S_n, see Definition 5, while the following steps correspond to counting the number of non-zero entries of the matrix associated to these permutations (there are at most $2n+1$ of them since the system counts to at most n and each count takes two steps; one extra step may be needed for a technical reasons: to reset to 0 the polarization of membranes that had polarization 1 after the first Nn steps).

It should be noted that the requirement that the output region is the environment (typically done for decisional problem solutions) has been dropped. This makes it possible to give non-polynomial answers to the permanent problem (which is a number between 0 and $n!$) in a polynomial number of steps without having to recall from [1] rules sending objects out that work in parallel.

4 Attacking PP Complexity Class

The probabilistic polynomial complexity class **PP**, also called Majority-P, has been introduced in [2]. It is the class of **decision** problems solvable by a probabilistic Turing machine in polynomial time, with an error probability of less than 1/2 for all instances, see also [14]. It is known that $\mathbf{PP} \supseteq \mathbf{NP} \cup \mathbf{co} - \mathbf{NP}$, and the inclusion is strict if $\mathbf{P} \neq \mathbf{NP}$. Therefore, showing a solution to a **PP**-complete problem by P systems with active membranes without division of non-elementary membranes and without membrane creation would improve the best known results relating P systems to $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP}$.

In this section we show a way to do this, paying a small price of post-processing. We recall that the framework of solving decisional problems by P systems with active membranes includes two encoding functions (computing the description of a P system from the size of the problem instance and computing the input multiset from the instance of the problem). Unlike a more general case of solving computational problems, there was no need for the decoding function, since the meaning of objects `yes` and `no` sent to the environment was linked with the answer. While the decoding function was necessary for extending the framework for the computational problems (computing the answer to the instance of the problem from the output multiset of a P system in polynomial time), we would like to underline that it is useful even for the decisional problems.

It is not difficult to see that the problem "given a matrix A of size n, is $Perm(A) > n!/2$?" is **PP**-complete. Hence, we only have to compare the result of the computation of the matrix permanent with $n!/2$. Doing it by usual P systems with active membranes would need a non-polynomial number of steps. We can propose two approaches.

- Generalizing rules of type (a) to cooperative ones. It would then suffice to generate $n!/2$ copies of a new object z, then erase pairs of o and z and finally check if some object o remains. However, this class of P systems is not studied.
- Consider, as before, the number of objects o as the result of the computation of a P system. Use the decoding function

$$dec(x) = \begin{cases} \texttt{no} & , x \leq n!/2, \\ \texttt{yes} & , x > n!/2. \end{cases}$$

  The function dec can obviously be computed in polynomial time.

5 Discussion

In this paper we presented a solution to the problem of computing the permanent of a binary matrix by P systems with active membranes, namely with two polarizations and rules of object evolution, sending objects out and membrane division. This problem is known to be **#P**-complete. The solution has been preceded by the framework that generalizes decisional problems to computing

functions: now the answer is much more than one bit. This result suggests that P systems with active membranes without non–elementary membrane division still compute more than decisions of the problems in $\mathbf{NP} \cup \mathbf{co} - \mathbf{NP}$. Indeed, paying the small price of using the decoding function also for decisional problem this approach allows to solve problems in the class **PP**, which is strictly larger than that (assuming $\mathbf{P} \neq \mathbf{NP}$).

Acknowledgments. All authors gratefully acknowledge the support by the Science and Technology Center in Ukraine, project 4032. Yurii Rogozhin gratefully acknowledges the support of the European Commission, project MolCIP, MIF1-CT-2006-021666.

References

1. Alhazov, A., Pan, L., Păun, G.: Trading polarizations for labels in P systems with active membranes. Acta Informaticae 41, 111–144 (2004)
2. Gill, J.: Computational complexity of probabilistic Turing machines. SIAM Journal on Computing 6, 675–695 (1977)
3. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A fast P system for finding a balanced 2-partition. Soft Computing 9, 673–678 (2005)
4. Păun, G.: P systems with active membranes: Attacking NP–complete problems. J. Automata, Languages and Combinatorics 6, 75–90 (2001)
5. Păun, G., Suzuki, Y., Tanaka, H., Yokomori, T.: On the power of membrane division in P systems. Theoretical Computer Sci. 324, 61–85 (2004)
6. Pérez-Jiménez, M.J., Riscos-Núñez, A.: Solving the subset-sum problem by active membranes. New Generation Computing 23, 367–384 (2005)
7. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: Complexity classes in cellular computing with membranes. Natural Computing 2, 265–285 (2003)
8. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: Computationally hard problems addressed through P systems. In: Ciobanu, G., et al. (eds.) Applications of Membrane Computing, pp. 315–346. Springer, Heidelberg (2006)
9. Pérez Jiménez, M.J., Romero Campero, F.J.: Attacking the common algorithmic problem by recognizer P systems. In: Margenstern, M. (ed.) MCU 2004. LNCS, vol. 3354, pp. 304–315. Springer, Heidelberg (2005)
10. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Sci. 8, 189–201 (1979)
11. Wegener, I.: Complexity Theory: Exploring the Limits of Efficient Algorithms. Springer, Heidelberg (2005)
12. Williams, R.M., Wood, D.H.: Exascale computer algebra problems interconnect with molecular reactions and complexity theory. DIMACS Series in Discrete Mathematics and Theoretical Computer Sci. 44, 267–275 (1999)
13. `http://en.wikipedia.org/wiki/Permanent` (updated 05.05.2008)
14. `http://en.wikipedia.org/wiki/PP_complexity` (updated 09.09.2008)
15. `http://en.wikipedia.org/wiki/Sharp-P` (updated 13.12.2007)