Appl. Math. Inf. Sci. **7**, No. 2L, 549-553 (2013)

Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/072L26

Arithmetic Expression Evaluation by P Systems

Ping Guo[1,3], *Hai-Zhu Chen*[2]

[1]College of Computer Science, Chongqing University, Chongqing 400030, China
[2]Department of Software Engineering, Chongqing College of Electronic Engineering,Chongqing 401331, China
[3]Chongqing Key Laboratory of Software Theory & Technology, Chongqing 400044, China

Received: 13 Oct. 2012, Revised: 24 Jan. 2013, Accepted: 27 Jan. 2013
Published online: 1 Jun. 2013

Abstract: Arithmetic operations and expression evaluations are fundamental in computing models. We firstly designs a family of P systems for arithmetic operations (called arithmetic P systems) and gives the rules without priority. According to the arithmetic expression,an expression P system can be constructed through the algorithm and transmission rules proposed in this paper. Arithmetic P systems are one-membrane systems and several of them can make up an expression P system, where arithmetic operations can be performed in parallel and the computing results can be transmitted under the control of the transmission rules. Then a computing architecture can be built and based on it different expression P systems can be constructed by designing different parallel strategies.

Keywords: Membrane computing, cell-like P systems, arithmetic operations, arithmetic P systems, expression P systems

1 Introduction

Membrane computing (also called P systems) is an emerging branch of nature computing since introduced by Păun in 1998 as distributed parallel computing model [1]. It is aiming to abstract computing models from the structures and the functions of living cells and from their interactions in tissues or higher order biological structures. After Păun proposed and proved that P system based on membrane division can solve SAT problem in polynomial time [2], many variants of P systems including tissue-like and neural-like ones have been successfully used to design solutions to NP-complete problems such as SAT [3], Subset- Sum [4], HPP [5] and 2-partition [6].

The researches on arithmetic operations in P system are relatively weak compared to the ones in solving NP-complete problem. Atanasiu firstly constructs arithmetic P systems with operands encoded in base two [7]. Ciobanu builds arithmetic P systems with operands encoded by a simple and natural encoding [8]. Literature [9] design such P systems without priority rules and multi-membrane P systems are constructed for signed operands [10]. Taking neural-like P systems as computing devices can be found in literature [11].

The research on P systems for arithmetic expression evaluation attracts few attentions. Literature [12] studies such P systems with priority rules. In this paper, we construct arithmetic P system without priority rules firstly. Then we propose an algorithm to build an expression P system according to the arithmetic expression and design transmission rules subsequently. Arithmetic P system is one-membrane system. When several of such systems are integrated into an expression P system, the arithmetic operations defined in each membrane can be performed in parallel and the computing results can be transmitted under the control of the transmission rules. Then a computing architecture can be built. Our works makes the P system for evaluating arithmetic expression easier to be implemented. In the rest of this paper, a family of arithmetic P systems without priority rules are constructed in Section 2 and expression P systems are built in Section3 with the constructing algorithm and transmission rules presented in detailed.

2 Arithmetic P Systems

According to the definition of cell-like P systems [1], an arithmetic P system based on single membrane can be defined as:

$$\Pi = (O, w, R^a) \tag{1}$$

where
1) O is an finite and non-empty alphabet of objects;

* Corresponding author e-mail: guoping@cqu.edu.cn

2) w is a multiset over O, and it is a set of objects in the membrane of Π;
3) R^+, R^-, $R^\times$ and $R^\div$ are rule sets for implementing addition, subtractions, multiplication and division respectively and the rule set for the family of arithmetic P system is $R^a = R^+ \cup R^- \cup R^\times \cup R^\div$. Fig.1 illustrates the structure of arithmetic P system with an instance shown in Fig.2.

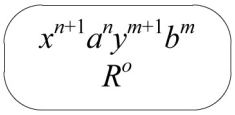

Fig. 1 The structure of arithmetic P systems ($o \in \{+,-,\times,\div\}$)

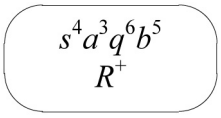

Fig. 2 The arithmetic P system for $3+(-5)$

We have some conventions as follows:
1) The cardinalities of objects a and b denote the absolute values of the first and second operands respectively while objects x and y represent the signs of the operands respectively;
2) R^o is a set of rules, $o \in \{+,-,\times,\div\}$;
3) $x \in \{s,p\}$, $x = s$ means that the first operand is positive while $x = p$ means negative. For the second operand, the meaning of $y \in \{t,q\}$ is similar to the one of x;
4) After the computation is finished, the cardinality of r is the absolute value of the result with object $+$ representing nonnegative while object $-$ representing negative;
5) All of the available rules must be applied in a time slice and some rule can be applied for several times in a time slice in a maximal parallel way.

The rules in R^a will be discussed in detailed as follows.

2.1 Addition

The rules for addition are designed as:

$$R^+ = R_1^+ \cup R_2^+ \cup R_3^+ \cup R_4^+ \cup R_5^+ \tag{2}$$

where,

$$\begin{aligned}
R_1^+ &= \{r_1 : c \to de, r_2 : ad \to a, r_3 : bd \to b,\\
&\quad r_4 : e \to xy, r_5 : dx \to +\}\\
R_2^+ &= \{r_6 : sta \to +rt^2, r_7 : stb \to +rs^2, r_8 : sty \to +st\}\\
R_3^+ &= \{r_9 : sqab \to \lambda, r_{10} : sqya \to +^2ry^2q^2,\\
&\quad r_{11} : sqyb \to -^2ry^2s^2\}\\
R_4^+ &= \{r_{12} : ptab \to \lambda, r_{13} : ptya \to -^2ry^2t^2,\\
&\quad r_{14} : ptyb \to +^2ry^2p^2\}\\
R_5^+ &= \{r_{15} : pqa \to -rq^2, r_{16} : pqb \to -rp^2,\\
&\quad r_{17} : pqy \to -pq, r_{18} : pq+ \to -pq\}
\end{aligned}$$

$R_1^+ \cup R_2^+$ is responsible for the addition of two nonnegative operands while $R_1^+ \cup R_3^+$ for nonnegative operand added by the negative one, $R_1^+ \cup R_4^+$ for negative operand added by the nonnegative one, and $R_1^+ \cup R_5^+$ for the addition of two negative operands. The sum is composed by the cardinality of objects r and the object $+$ or $-$ in the membrane.

For example, the procedure of the addition of two nonnegative operands is:
1) time-slice=1, $\{r_1, r_6, r_7\}$ is a set of available rules;
2) time-slice=2, $\{r_2, r_3, r_4, r_6, r_7\}$ is a set of available rules;
3) time-slice=3, $\{r_5, r_6, r_7, r_8\}$ is a set of available rules;
4) from time-slice=4, $\{r_6, r_7, r_8\}$ is a set of available rules in every time-slice.

The sum of 5 and -8 can be obtained by $R_1^+ \cup R_3^+$. Table1 shows the cardinalities of objects in Π^+ at each time slice during the computation. In the last row of Table 1, the cardinality of r is 3 and the one of $-$ is 6, so we have -3. The complexity of the operations in Π^+ is $O(1)$.

Table 1 The example of Π^+ applying to $5+(-8)$

Time slice	a	b	c	d	e	q	r	s	x	y	$-$	rules
0	5	8	1	0	0	9	0	6	0	0	0	
1	0	3	0	1	1	4	0	1	0	0	0	r_1,r_9
2	0	3	0	0	0	4	0	1	1	1	0	r_3,r_4
3	0	2	0	0	0	3	1	2	1	2	2	r_{11}
4	0	0	0	0	0	1	3	4	1	4	6	r_{11}

2.2 Subtraction

Similarly to addition, the rules for subtraction can be designed as:

$$R^- = R_1^- \cup R_2^- \cup R_3^- \cup R_4^- \cup R_5^- \tag{3}$$

where,

$$\begin{aligned}
R_1^- &= \{r_1 : c \to de, r_2 : ad \to a, r_3 : bd \to b,\\
&\quad r_4 : e \to xy, r_5 : dx \to +\}\\
R_2^- &= \{r_6 : stab \to \lambda, r_7 : stya \to +^2ry^2t^2,\\
&\quad r_8 : styb \to -^2ry^2s^2\}\\
R_3^- &= \{r_9 : sqa \to +rq^2, r_{10} : sqb \to +rs^2,\\
&\quad r_{11} : sqy \to +sq\}\\
R_4^- &= \{r_{12} : pta \to -rt^2, r_{13} : ptb \to -rp^2,\\
&\quad r_{14} : pty \to -pt, r_{15} : pt+ \to -pt\}
\end{aligned}$$

$$R_5^- = \{r_{16} : pqab \to \lambda, r_{17} : pqya \to -^2ry^2q^2, r_{18} : pqdb \to +^2ry^2p^2\}$$

In these rules, $R_1^- \cup R_2^-$ is responsible for the subtraction of two nonnegative operands while $R_1^- \cup R_3^-$ for nonnegative operand subtracted by the negative one, $R_1^- \cup R_4^-$ for negative operand subtracted by the nonnegative one, and $R_1^- \cup R_5^-$ for the subtracted of two negative operands. So we obtain the set of rules for subtraction. Similarly to Π^+, the complexity of subtraction in Π^- is $O(1)$.

2.3 Multiplication

We decide the sign of the product by the two operands directly and then multiply the absolute values of them to get the absolute value of the product. The rules for deciding the sign of the product are:

$$R_1^\times = \{r_1 : stc \to +, r_2 : sqc \to -, r_3 : ptc \to -, r_4 : pqc \to +\}$$

And the rules for computing the absolute value of the product are:

$$R_2^\times = \{r_5 : ab \to ijkr\}$$
$$R_3^\times = \{r_6 : aje \to juv, r_7 : bje \to juv, r_8 : i \to z, r_9 : kd \to w\}$$
$$R_4^\times = \{r_{10} : jze \to xu, r_{11} : jzv \to y, r_{12} : jzk \to y\}$$
$$R_5^\times = \{r_{13} : uw \to ed, r_{14} : x \to b, r_{15} : y \to ab\}$$

So we have the rule set for multiplication:

$$R^\times = R_1^\times \cup R_2^\times \cup R_3^\times \cup R_4^\times \cup R_5^\times \quad (4)$$

$R_2^\times$, $R_3^\times$, $R_4^\times$ and $R_5^\times$ are mutually exclusive, namely, only one of them can be applied at each time slice. What's more, the rules in every one of them must be applied in a maximal parallel way.

At the end of the multiplication, the absolute value of the product is the cardinality of r. The product is nonnegative if the cardinality of $+$ is more than one, otherwise it is negative. At worst, the complexity of multiplication in $\Pi^\times$ is $O(n^2)$.

The product of 4 and 2 can be obtained by $R^\times$ and the result is +8 according to the last row in Table 2. $l_i (1 \le i \le 5)$ denotes the i-th loop in applying rules in $R^\times$ to the example.

Table 2 The example of $\Pi^\times$ applying to 4×2)

	a	b	c	d	e	p	q	r	s	t	$+$	$-$
l_0	4	2	1	1	1	0	0	0	5	3	0	0
l_1	3	2	0	1	1	0	0	2	4	2	1	0
l_2	2	2	0	1	1	0	0	4	4	2	1	0
l_3	1	2	0	1	1	0	0	6	4	2	1	0
l_4	1	1	0	1	1	0	0	7	4	2	1	0
l_5	0	1	0	1	1	0	0	8	4	2	1	0

2.4 Division

Similar to multiplication, we decide the sign of the quotient by the two operands directly and then compute the division of the absolute values of them to get the absolute value of the quotient. The rules for deciding the sign of the quotient are:

$$R_1^\div = \{r_1 : stc \to +, r_2 : sqc \to -, r_3 : ptc \to -, r_4 : pqc \to +\}$$

And the rules for computing the absolute value of the quotient are:

$$R_2^\div = \{r_5 : a \to e, r_6 : z \to uw, r_7 : gb \to b\}$$
$$R_3^\div = \{r_8 : eb \to ij, r_9 : gu \to f, r_{10} : w \to v\}$$
$$R_4^\div = \{r_{11} : eiu \to red, r_{12} : biu \to bd, r_{13} : j \to x, r_{14} : v \to k\}$$
$$R_5^\div = \{r_{15} : uik \to rd, r_{16} : x \to y\}$$
$$R_6^\div = \{r_{17} : dy \to buw, r_{18} : iy \to b\}$$

So we have the rule set for division:

$$R^\div = R_1^\div \cup R_2^\div \cup R_3^\div \cup R_4^\div \cup R_5^\div \cup R_6^\div \quad (5)$$

In $R_1^\div$ and $R_2^\div$, the rules which can be applied are applied firstly ($R_1^\div$ is used to get the sign of the quotient and $R_2^\div$ is used to prepare for the applications of $R_3^\div \sim R_6^\div$ and then we apply the rules in $R_3^\div \sim R_6^\div$). $R_3^\div$, $R_4^\div$, $R_5^\div$ and $R_6^\div$ are mutually exclusive, namely, only one of them can be applied at each time slice. What's more, the rules in every one of them must be applied in a maximal parallel way.

At the end of the division, the cardinality of r is the result, namely, the whole-number part of the division $n \div m$, and it is nonnegative if the cardinality of $+$ is more than one, otherwise it is negative. Particularly, appearance of object f means that the result is overflown (namely, $m = 0$). It is interesting to see that the complexity of division in $\Pi^\div$ is $O(n/m)$.

3 Expression P Systems

3.1 Construction of expression P systems

To use the rules proposed in Section 2 to evaluate arithmetical expression, we present an algorithm to construct expression P system.

Algorithm: constructing expression P systems
Input: arithmetical expression E
Output: expression P system M
step1:transform E into suffix forms and still name it E
step2: generate a stack S to store M, and initialize S to be void
step3: process the symbols of E in turn, name the current symbol to be processed as x
If x is operand then
Begin
construct a membrane not including any object, and name it as M, put the transition rules (pre-

sented later) into M
put object r into M (the cardinality of r is $|x|$)
If $x \geq 0$
put an object $+$ into M
Else
put an object $-$ into M
put an object c in to M
push M into S
End
Else
Begin
pop off the top of S, name it as M_2, and put an object $'2'$ into M_2
pop off the top of S, name it as M_1, and put an object $'1'$ into M_1
construct a membrane not including any object, and name it as M
find out the rules corresponding to x from Section 2 and put them into M
put M_1 and M_2 into M and take them as inner membranes of M
put symbol c into M
put the transition rules (presented in the next subsection) into M
push M into S
End
step4: pop off the top of S, name it as M, and put an object $'1'$ into M
step5: Return M

Usually, the infix form of arithmetical expression is familiar to us, for example,

$$(3-(-2)) \times 3+5 \times (4 \div 2)$$

According this algorithm, the expression P system for evaluating it can be constructed as shown in Fig.3.

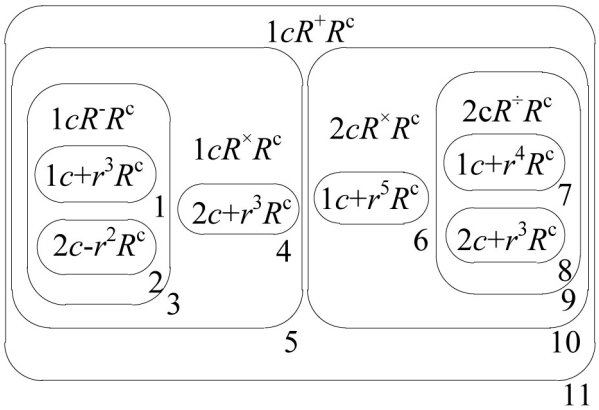

Fig. 3 The expression P system for evaluating $(3-(-2)) \times 3+5 \times (4 \div 2)$

In Fig.3, object c placed in every membrane is used to trigger the transmission rules.

3.2 Transmission rules

The result in current membrane is sent into the outer membrane by transmission rules. According to the rule set $R^o(o \in +,-,\times,\div)$ in Section 2, transmission rules, called R^c, will be designed according to the four cases:
1) The result in the current membrane is nonnegative and it will be the first operand in the outer membrane, object s and a will be sent into the outer membrane;
2) The result in the current membrane is negative and it will be the first operand in the outer membrane, object p and a will be sent into the outer membrane;
3) The result in the current membrane is nonnegative and it will be the second operand in the outer membrane, object t and b will be sent into the outer membrane;
4) The result in the current membrane is negative and it will be the first operand in the outer membrane, object q and b will be sent into the outer membrane.

R^c is composed of five parts:

$$R^c = R_1^c \cup R_2^c \cup R_3^c \cup R_4^c \cup R_5^c \tag{6}$$

Where,

$$\begin{aligned}
R_1^c &= \{r_1 : \gamma \to \varepsilon, r_2 : \varepsilon\tau \to \omega, r_3 : \varepsilon\omega \to (), out)\} \\
R_2^c &= \{r_4 : 1c+ \to \alpha^2\mu^2\gamma^2\omega(s,out), \\
&\quad r_5 : \alpha\mu\omega r \to \tau(sa,out), \\
&\quad r_6 : \alpha\mu\varepsilon r \to \alpha^3\mu^3\gamma^3\omega(sa,out)\} \\
R_3^c &= \{r_7 : 1c- \to \alpha^2\eta^2\gamma^2\omega(p,out), \\
&\quad r_8 : \alpha\eta\omega r \to \tau(pa,out), \\
&\quad r_9 : \alpha\eta\varepsilon r \to \alpha^3\eta^3\gamma^3(pa,out)\} \\
R_4^c &= \{r_{10} : 2c+ \to \beta^2\zeta^2\gamma^2\omega(t,out), \\
&\quad r_{11} : \beta\zeta\omega r \to \tau(tb,out), \\
&\quad r_{12} : \beta\zeta\varepsilon r \to \beta^3\zeta^3\gamma^3(tb,out)\} \\
R_5^c &= \{r_{13} : 2c- \to \beta^2\varphi^2\gamma^2\omega(q,out), \\
&\quad r_{14} : \beta\varphi\omega r \to \tau(qb,out), \\
&\quad r_{15} : \beta\varphi\varepsilon r \to \beta^3\varphi^3\gamma^3(qb,out)\}
\end{aligned}$$

In R^c, $R_1^c \cup R_2^c$ are used to send the objects that represent the nonnegative result into the outer membrane as the first operand; $R_1^c \cup R_3^c$ are used to send the objects that represent the negative result into the outer membrane as the first operand; $R_1^c \cup R_4^c$ are used to send the objects that represent the nonnegative result into the outer membrane as the second operand; and $R_1^c \cup R_5^c$ are used to send the objects that represent the negative result into the outer membrane as the second operand.

The rules in expression P systems are:

$$R = R^a \cup R^c = R^+ \cup R^- \cup R^\times \cup R^\div \cup R^c \tag{7}$$

3.3 Procedure of evaluating arithmetic expressions

We explain the procedure of the evaluation of arithmetic expression through the example shown in Fig.3. According to Fig.3 and the rules presented previously, the parallel evaluation of the expression can be described as:
1) membranes 1, 2, 4, 6, 7, 8 apply the rules in R^c in

parallel to pre-process the operands and send out the results.
2) membrane 3 applies the rules in R^- to compute $3-(-2)$, similarly, membrane 9 processes $4 \div 2$;
3) membrane 3 and 9 use the rules in R^c to send out the results obtained in 2) respectively;
4) membrane 5 applies the rules in $R^\times$ to compute 5×3(5 comes from $3-(-2)$) and saves the result, similarly, membrane 10 processes 5×2(2 comes from $4 \div 2$);
5) membrane 5 and 10 use the rules in R^c to send out the results obtained in 4) respectively;
6) membrane 11 applies the rules in R^+ to compute $15+10$(15 and 10 come from 5×3 and 5×2 respectively);
7) membrane 11 uses the rules in R^c to sends the result into the environment;
8) the whole computation is finished.

In the above procedure of the evaluation, the synchronization is required besides the parallelism. For this example, the synchronization include:
1) the computation in membrane 3 can be triggered once a part of results sent by membranes 1 and 2 arrive;
2) the computation in membrane 9 will be triggered when all of the results sent by membranes 7 and 8 arrive;
3) the computation in membrane 5 will be triggered when all of the results sent by membranes 3 and 4 arrive;
4) the computation in membrane 10 will be triggered when all of the results sent by membranes 6 and 9 arrive;
5) the computation in membrane 11 can be triggered once a part of results sent by membranes 5 and 10 arrive.

So we can see that different rules for arithmetic operations require different synchronization in expression P systems.For realizing the actual computations,the parallelism strategies, namely synchronization and mutual exclusion among the rules in R^a and R^c, will be studied in the further work.

4 Conclusions

In this paper, we propose arithmetic P systems without priority rules based on cell-like P systems. We propose an algorithm to construct expression P systems according to arithmetical expressions and the design transmission rules. Expression P systems are composed by several arithmetical P systems where arithmetical operations can be performed in parallel and the computing results can be transmitted under the control of the transmission rules. Based on the rules and different parallel strategies, we can construct and realize different P systems for evaluating arithmetic expressions. The future researches will focus on the parallelism strategies among the rules and the P system for the arithmetical operations of real operands.

Acknowledgement

This work is supported by Natural Science Foundation Project of CQ CSTC (No.cstc2012jjA40022).

References

[1] Gh. Păun, Journal of Computer and System Science **61**, 108-143 (2000).
[2] Gh. Păun, Proc. International Conference on Unconventional Models of Computation, Springer, London, 94-115 (2000).
[3] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, International Journal of Computers, Communications & Control **3**, 295-303 (2008).
[4] A. Leporati, M.A. Gutiérrez-Naranjo, Fundamenta Informaticae **87**, 61-77 (2008).
[5] L. Pan, A. Alhazov, Acta Informatica **43**, 131-145(2006).
[6] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Soft Computing **9**, 673-678 (2005).
[7] A. Atanasiu, Romanian Journal of Information Science and Technology **4**, 5-20 (2001).
[8] G. Ciobanu, International Journal of Computers, Communications & Control **1**,13-24 (2006).
[9] P. Guo, J. Chen, Proc. International Conference on BioMedical Engineering and Informatics, 231-234 (2008).
[10] P. Guo, M. Luo, Proc. International Conference on Information Science and Engineering, 393-396 (2009).
[11] M.A. Gutiérrez-Naranjo, A. Leporati, International Journal of Computers, Communications & Control **4**, 244-252 (2009).
[12] P. Guo, S.J. Liu, Advanced Materials Research **225-226**, 1115-119 (2011).

Ping Guo received his PhD (2004) in Computer software and theory from Chongqing University. Now he is a full professor in College of Computer Science, Chongqing University, China. His current research interests include different aspects of Artificial Intelligence and Biological computing model. He has (co-)authored 1 books and more than 130 papers.

Hai-zhu Chen received her M.Sc. (2005) and PhD (2011) in Computer science and technology from Chongqing University. Now she is a lecturer at Software Engineering Department, Chongqing College of Electronic Engineering, China. Her current research interests include membrane computing and optimization algorithm. She has (co-)authored more than 10 papers.