PARSING BY SIMPLE INSERTION SYSTEMS

Gemma Bel-Enguix[1], Pál Dömösi[2] and Alexander Krassovitsky[1]

[1]*GRLMC, Rovira i Virgili University, Avda. Catalunya 35, 43002 Tarragona, Spain*
[2]*College of Nyíregyháza, Institute of Mathematics and Informatics*
H-4400 Nyíregyháza, Sóstói út 31/B, Hungary

Keywords: Insertion systems, Parsing, Natural language, Multi-agent systems.

Abstract: The aim of this paper is to initiate a new direction for the investigation of multi-agent systems. We will consider the insertion systems as very simple multi-agent systems, where the agents are consisting of their insertions. We define the systems and describe their working and main features. The central develoment of the paper is the application of such systems to parsing. Some examples to natural language processing are introduced that can illustrate the system.

1 INTRODUCTION

Although interaction between the fields of formal languages and multi-agents systems is not frequent, there are some examples that illustrate the high theoretical and applied potential of such collaboration.

Grammar Systems (Csuhaj-Varjú et al., 1994) are widely considered as a particular case of multi-agent system focusing in the special task of generating – and accepting (Fernau et al., 1996)– languages.

Another interesting interdisciplinar approach was given by Networks of Evolutionary Processors (NEPs), introduced in (Castellanos et al., 2003), a type multi-agent systems in the area of formal languages and bio molecular computing. An overview on the generative power and complexity results of NEPs has can be found in (Martín-Vide and Mitrana, 2005).

It is not easy to find in the literature practical applications of Grammar Systems or NEPs. In what refers to Networks of Evolutionary Processors, some ideas have been launched for parsing of natural languages (Bel-Enguix et al., 2009; Ortega et al., 2009) starting from the idea of accepting NEPs, introduced in (Margenstern et al., 2004) and developed in several papers (Manea and Mitrana, 2009).

In this work, we will describe a polynomial parser working on insertion (derivation) systems which can be considered as very simple multi-agent systems. Following with the tradition of NEPs, we want to use a method that can make some contribution to both multi-agent systems and formal languages. We are also preliminary applying the mechanism to parsing of natural languages.

2 INSERTION SYSTEMS

Insertion systems have been introduced and studied in (Galiukschov, 1981). Characterization of recursively enumerable languages by insertion systems is given in (Păun et al., 1998; Kari and Sosík, 2009; Madhu et al., 2009).

An *insertion system* (which is also called insertion grammar in the literature) is a construct $\mathcal{G} = (V, A, P)$, where V is the (finite) *alphabet*, $A \subset V^*$ is the (finite) set of *axioms*, and $P \subset \{(u, \alpha, v) \mid u, \alpha, v \in V^*$ is the (finite) set of *insertion* rules. An insertion rule $(u, \alpha, v) \in I$ indicates that the string α can be inserted in between u and v. The rule $(u, \alpha, v) \in I$ corresponds to the rewriting rule $uv \Longrightarrow u\alpha v$. We denote by $\Longrightarrow$ the relation defined by an insertion rule (formally, $x \Longrightarrow y$ iff $x = x_1uvx_2, y = x_1u\alpha vx_2$, for some $(u, \alpha, v) \in I$ and $x_1, x_2 \in V^*$). We denote by $\Longrightarrow^*$ the reflexive and transitive closure of $\Longrightarrow$ (as usual, $\Longrightarrow^+$ is its transitive closure). The language generated by $\mathcal{G}$ is defined by

$$L(\mathcal{G}) = \{w \in V^* \mid x \Longrightarrow^* w, x \in A\}.$$

We say that an insertion system has weight

Bel-Enguix G., Dömösi P. and Krassovitskiy A. (2010).
PARSING BY SIMPLE INSERTION SYSTEMS.
In *Proceedings of the 2nd International Conference on Agents and Artificial Intelligence*, pages 383-387
DOI: 10.5220/0002784803830387

(l,m,m') if

$$l = max\{|\alpha| \mid (u,\alpha,v) \in P\};$$
$$m = max\{|u| \mid (u,\alpha,v) \in P\};$$
$$m' = max\{|v| \mid (u,\alpha,v) \in P\}.$$

For example, consider an insertion system of the weight $(2,0,0)$ $\Pi = (\{a,b\},\{\varepsilon\},\{(\varepsilon,ab,\varepsilon)\})$. Then, it is clear that Π generates the Dyck language.

We denote by $INS_l^{m,m'}$ the family of languages generated by insertion systems of weight (l,m,m'). In the sequels we will restrict ourselves to insertion systems of weight $(l,1,1)$, i.e., we assume that all insertions have the form $(a,a_1\cdots a_k,b)$, where $a,a_1,\ldots,a_k,b \in V$, and $k \leq l$. It is well-known that the family of languages $INS_l^{1,1}, l \geq 1$ is a proper subfamily of context-free languages (Păun, 1997), for which there are well-known parsers working in polynomial time. Therefore, we expect to find parsers of polynomial power for insertion systems of weight $(l,1,1)$ too.

3 PARSING FOR SIMPLE INSERTION SYSTEMS

For the sake of simplicity, first we assume that A is a singleton having a string of length two. We call this type of insertion systems *simple*.

For simple insertion systems, we can show an effective parsing technique based on the idea of CYK algorithm (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967). The input to the algorithm is a simple insertion system $\mathcal{G} = (V,A,P)$ and a string $w = a_1\cdots a_n \in V^*$. In $O(n^3)$ time, the algorithm constructs a table that checks whether w is in $L(\mathcal{G})$. Note that when computing the running time of the checking, the system itself is considered fixed, and its size contributes only a constant factor to the running time, which is measured in terms of the length of the string w whose membership in $L(\mathcal{G})$ is being tested. In this algorithm, we construct a triangular table, called parsing table, as shown in Table 1.

The parsing table consists of $n-1$ rows and $n-1$ columns, where n denotes the length of the parsed word. To fill the table, we work row-by-row upwards. Each row corresponds to substrings of the given length; the bottom row is for strings of length 2, the second row for strings of length 3, etc., until the top row corresponds to the only substring of length n, which is the parsed word. By the proposed algorithm it takes $O(n)$ time to compute each entry of the table. Since there are $n(n-1)/2$ entries, the whole table construction takes $O(n^3)$ time.

Table 1: Parsing Table.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
a_1	a_2	a_3	a_4	a_5

The Algorithm

1. If a_1a_m is not the (only) axiom (i.e. $\{a_1a_m\} \neq A$) then $a_1\cdots a_m \notin L(\mathcal{G})$ and we are ready.
2. Otherwise, we consider the following treatment.
   (a) for every $j = 1,\ldots,m-1$, we put $(j,j+1)$ into $X_{j,j+1}$.
   (b) for every $i = 2,...,m-1$ and $j = 1,...,m-i$, we put $(j,i+k)$ into $X_{j,i+k}$ if we did not do it so far in the previous steps, and there exists an insertion $(a_j,a_{j_1}a_{j_2}\cdots a_{j_s},a_{i+k})$ $(1 \leq j < j_1 < j_2 < \cdots < j_s < i+k \leq m)$ such that $(j,j_1) \in X_{j,j_1},(j_1,j_2) \in X_{j_1,j_2},\ldots,(j_{s-1},j_s) \in X_{j_{s-1},j_s},(j_s,i+k) \in X_{j_s,i+k}$ [1]
   (c) Finally, if $(a_1,a_m) \in X_{1,m}$ then $a_1\cdots a_m \in L(\mathcal{G})$, otherwise not.

Prove that the algorithm works in polynomial ($O(n^3)$) time. Moreover, for every string $a_1\cdots a_m \in V^*$, we have $a_1\cdots a_m \in L(\mathcal{G})$ if and only if $(a_1,a_m) \in X_{1,m}$.

Theorem 1. Let Π be a simple insertion systems and let n be the length of the parsed word. There exists a parser working on Π in polynomial ($O(n^3)$) time.

Proof:

The reason the algorithm puts the correct pairs of characters is the following. In the bottom row, which has the length $n-1$, for every position we put a pair consisting of the line number of the position and the consecutive one. Thus, the first pair is $(1,2)$ and the last pair is $(n-1,n)$.

Then, for every $i = 2,...,m-1$ and $j = 1,...,m-i$, we can put (s,t) into the j-th position of the i-th row if and only if two conditions are holding: a.) $s = j$; b.) there exist an insertion $(a_j,a_{j_1}a_{j_2}\cdots a_{j_s},a_{i+1})$ and positions $X_{j,j_1-1},X_{j_1,j_2-1},\ldots,X_{j_{s-1},j_s-1},X_{j_s,j_{i+k}}$ which

[1] Note that, by our assumptions, $|a_{j_1}a_{j_2}\cdots a_{j_s}| \leq \ell$. Therefore, the number of elementary operations in step 2 is not more that $(m-2)(m-i)\ell$.

contain pairs of line numbers of the parsed string such that their column index is the same as the first member of the contained pair, and their first indexes form a sequence $j_1-1, j_2-1, \ldots, j_s-1$, where $j_1, \ldots, j_s$ are the indexes of letters in the parsed strings for which $a_{j_1}a_{j_2}\cdots a_{j_s}$ is the middle part of the insertion rule $(a_j, a_{j_1}a_{j_2}\cdots a_{j_s}, a_{i+k})$.

For the running time, note that there are $O(n^2)$ to compute, and each involves comparing and computing with not more than $n\ell$ pairs of entries, where ℓ denotes the maximum of the length of the middle part in the insertion rules. We mention that the considered simple insertion system is fixed, and the number of its letters, its insertions do not depend on n, the length of the parsed string w. Thus, the time to compare at most ℓ positions is $O(1)$. As there are at most $n\ell$ such pairs for each position of the parsing table, the total working time is $O(n)$. Therefore, the running time of the parser is $O(n^3)$. □

4 PARSING FOR INSERTION SYSTEMS OF WEIGHT $(l,1,1)$

On the basis of our previous algorithm, we give a parser for general insertion systems of weight $(l,1,1)$ working also in polynomial time. Thus, the input to the algorithm is an insertion system $\mathcal{G} = (V,A,P)$ of weight $(l,1,1)$ and a string $w = a_1\cdots a_n \in V^*$.

Let us denote by ℓ the maximal length of axioms, and for every $1 \le i < i+k \le n$, denote by $\mathcal{G}_{i,i+k}$ the simple insertion system $\mathcal{G}_{i,i+k} = (V, \{a_i a_{i+k}\}, P)$. The algorithm works as follows. First we built a digraph with set of vertices $\{1,\ldots,n\}$ such that the pair $(i,j), 1 \le i < j \le n$ is an edge if there is a path from the vertex 1 to the vertex i having its length not more then $\ell-1$. Listing and checking the paths in this digraph, leading from 1 to n, we terminate the algorithm if one of the following conditions holds:

1. we found a path $1, i_1, \ldots, i_j, n$ with $a_1a_{i_1}\cdots a_{i_j}a_n \in L(\mathcal{G})$, or
2. for every path $1, i_1, \ldots, i_j, n$ such that it leads from the vertex 1 to the vertex i, we have $a_1a_{i_1}\cdots a_{i_j}a_n \notin L(\mathcal{G})$.

We establish $a_1a_{i_1}\cdots a_{i_j}a_n \overset{*}{\underset{G}{\Rightarrow}} a_1\cdots a_n$ (i.e., we establish $a_1\cdots a_n \in L(\mathcal{G})$ such that, we use consecutively our parser for simple insertion systems in order to show that $a_1a_{i_1} \overset{*}{\underset{G}{\Rightarrow}} a_1\cdots a_{i_1}, a_{i_1}a_{i_2} \overset{*}{\underset{G}{\Rightarrow}} a_{i_1}\cdots a_{i_2}$, …, $a_{i_j-1}a_{i_j} \overset{*}{\underset{G}{\Rightarrow}} a_{i_j-1}\cdots a_{i_j}, a_{i_j}a_n \overset{*}{\underset{G}{\Rightarrow}} a_{i_j}\cdots a_n$.

The Algorithm

1. The algorithm constructs a digraph $\mathcal{D}$ with set of vertices $\{1,\ldots,n\}$ in the following way.
   (a) let us label the edge 1 by 0.
   (b) For every $i = 2,\ldots,n$, Let $(1,i)$ be a (directed) edge of $\mathcal{D}$ if $(a_1,\ldots,a_i) \in L(\mathcal{G}_{1,i})$, [2], and in this case, let us label the edge i by 1.
   (c) For every $j = 3,\ldots,\ell$, $i = j,\ldots,n$ let (s,i) be an edge of $\mathcal{D}$ if s is labeled by $j-1$ and $(a_s, a_{s+1},\ldots,a_i) \in L(\mathcal{G}_{s,i})$, and in this case, let us label the edge i by j.
2. If the vertex n has no incoming edge, then $w \notin L(\mathcal{G})$, and we are ready.
3. Otherwise let us continue our treatment as follows.
   (a) Omit the vertices having no labels.
   (b) One after the other consider the paths $1, i_1, \ldots, i_j, n$ (which have less than ℓ length, leading from the edge 1 to the edge n in the reduced digraph), and check whether $a_1a_{i_1}\cdots a_{i_j}a_n$ is an axiom or not. If yes then $w \in L(\mathcal{G})$, and we are ready.
   (c) Running out of the paths such that nothing leads to an axiom, we can conclude $w \notin L(\mathcal{G})$, and we are ready again.

The system can have several interesting applications, like the use for parsing of natural language. Let us consider an example, with the following insertion rules:

(0) (¢,<boy> <eats> <cake>,$)

(1) (<a>,<very>,<very>)

(2) (<a>,<very>,<nice>)

(3) (<a>,<nice><young>,<boy>)

(4) (<a>,<nice><apple>,<cake>)

(5) (<¢>,<a>,<boy>)

(6) (<eats>,<a>,<cake>)

By this simple insertion system sentences like [1] can be parsed.

[1] ¢ a very very nice young boy eats a very nice apple cake $

The procedure is shown in Table 2.

Theorem 2. There exists a parser working on insertion systems of weigh $(l,1,1)$ in polynomial ($O(n^5)$) time.

[2] We can decide by the previously discussed parser whether it is true or not.

Table 2: An example of parsing by simple insertion system.

(1,14)													
(1,7)													
	(2,7)						(8,13)						
								(9,13)					
	(2,5)												
	(2,4)							(9,11)					
(1,2)	(2,3)	(3,4)	(4,5)	(5,6)	(6,7)	(7,8)	(8,9)	(9,10)	(10,11)	(11,12)	(12,13)	(13,14)	
¢	a	very	very	nice	young	boy	eats	a	very	nice	apple	cake	$
1	2	3	4	5	6	7	8	9	10	11	12	13	14

Proof: The correctness of the algorithm has been already explained. We built the digraph $\mathcal{D}$ in $\sum_{i=1}^{\ell} n-\ell = n(n-1)/2-(n-\ell)(n-\ell-1)/2$ elementary steps. Thus the number of elementary steps is $O(n^2)$. On the other hand, each of the elementary steps takes $O(n^3)$ running time. Thus the running time of the parser to built $\mathcal{D}$ is $O(n^5)$. The time to omit the non-labeled vertices is $O(n)$. Because the derived digraph is loop- and circle-free having not more than ℓ edges, the total number of the path in this digraph starting by 1 and finishing by n, is $2^{\ell-1}$. Recall that the considered insertion system with weight $(l,1,1)$ is also fixed and the number of its letters, its insertions, its axioms, and thus the maximum ℓ of the length of the axioms does not depend on n, the length of the parsed string. Therefore, the running time of the enumeration of the considered paths and checking whether the strings characterized by these paths are axioms or not, is $O(1)$. Therefore, the running time of the parser in total is $O(n^5)$. □

5 CONCLUSIONS

It is well-known that the family of languages, generated by insertion systems of weight $(l,1,1), l \geq 1$, is a proper subfamily of context-free languages (Păun, 1997), for which there are well-known parsers working in polynomial time. Therefore, it can be predicted that there are parsers of polynomial time complexity for languages generated by insertion systems of weight $(l,1,1)$ too. In this paper, we have introduced the concept of simple insertion systems and showed a parser running on the generated languages in time $O(n^3)$. On the basis of this result, we showed another parser for the languages generated by insertion systems working in $O(n^5)$ time. To find parsers of polynomial time complexity for languages generated by general insertion systems or, even by insertion systems of weight $(l,2,1)$ is not expected. It is shown in (Păun, 1997) that insertion systems of the weight $(2,2,2)$ can generate non-semililear languages. Hence, it is unlikely there is a polynomial time parser for these type of systems. For the future, it could be interesting to refine the presented parsers having lower time complexity (at least) for some family of languages generated by special insertion systems. Also, it is interesting to consider the parsing of insertion systems if an additional encoding of the output words (e.g. by the finite state transducer) is used. Some examples for application in natural language processing are also shown. It could be worthwhile to check the capacity of these systems and their future developments for parsing different syntactical structures.

ACKNOWLEDGEMENTS

This work has been supported by the project MTM2007-64322 from the Ministerio de Ciencia y Tecnología, and by the Active Researchers Program from the URV, Department of Romance Phylologies.

REFERENCES

Bel-Enguix, G., Jiménez-López, M. D., Mercas, R., and Perekrestenko, A. (2009). Networks of evolutionary processors as natural language parsers. In Filipe, J., Fred, A., and Sharp, B., editors, *Proceedings of the 1st International Conference on Agents and Artificial Intelligence*, pages 619–625. INSTICC Press.

Castellanos, J., Martín-Vide, C., Mitrana, V., and Sempere, J. M. (2003). Networks of evolutionary processors. *Acta Informatica*, 39:1–13.

Cocke, J. and Schwartz, T. (1970). Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York.

Csuhaj-Varjú, E., Dassow, J., Kelemen, J., and Păun, G. (1994). *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach, London.

Fernau, H., Holzer, M., and Bordihn, H. (1996). Accepting multi-agent systems: The case of cooperating distributed grammar systems. *Computers and artificial intelligence*, 15(2–3):105–264, 123–139.

Galiukschov, B. (1981). Semicontextual grammars (in russian). *Mat. Logica i Mat. Ling.*, pages 35–80.

Kari, L. and Sosík, P. (2009). On the weight of universal insertion systems. Manuscript.

Kasami, T. (1965). An efficient recognition and syntax-analysis algorithm for context-free languages. Technical Report AFCRL-65-758, Air Force, Cambridge Research Lab, Bedford, MA.

Madhu, M., Krithivasan, K., and Reddy, A. (2009). On characterizing recursively enumerable languages by insertion grammars. Technical report, III T, Hyberabad.

Manea, F. and Mitrana, V. (2009). Accepting networks of evolutionary processors. complexity aspects. In *Proceedings of the 1st International Conference on Agents and Artificial Intelligence*, pages 597–604. INSTICC Press.

Marcus, S. (1969). Contextual grammars. *Rev. Roum. Math. Pures Appl.*, 14:1525–1534.

Margenstern, M., Mitrana, V., and Pérez-Jiménez, M. (2004). Accepting hybrid networks of evolutionary processors. In *Pre-proceedings of DNA 10*, pages 107–117.

Martín-Vide, C. and Mitrana, V. (2005). *Networks of evolutionary processors: results and perspectives*, volume Molecular Computational Models: Unconventional Approaches, pages 78–114. Idea Group Publishing, Hershey.

Ortega, A., del Rosal, E., Pérez, D., Mercas, R., Perekrestenko, A., and Alfonseca, M. (2009). Pneps, neps for context free parsing: Application to natural language processing. *LNCS*, 5517:472–479.

Păun, G., Rozenberg, G., and Salomaa, A. (1998). *DNA Computing. New Computing Paradigms*. Springer-Verlag, Berlin.

Păun, G. (1997). *Marcus Contextual Grammars*. Kluwer, Dordrecht.

Younger, D. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.