Networks of Language Processors: a language theoretic approach to filtering and cooperation

Erzsébet Csuhaj-Varjú [*]
Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende u. 13-17
H-1111 Budapest
Hungary
E-mail: csuhaj@sztaki.hu

Abstract

Networks of language processors (NLP systems) is a collective term which has been introduced as a formal language theoretic framework for describing symbolic processing in highly (massively) parallel and distributed architectures. Roughly speaking, an NLP system consists of several language determining devices (language processors) which are located at nodes of a virtual graph (a network) and which rewrite strings and communicate them through the network. In this paper we briefly discuss the model and introduce a particular variant which can be considered as a formal model for collaborating agents which communicate with each other through a network and use recommendations for filtering information.

1 Introduction

One of the most challenging problems of current computer science is to develop sophisticated, highly reliable tools for supporting effective information dissemination and information search performed by users of computer networks. All who use Internet face similar questions every day: how to choose from the lot of information received and to be communicated, how to select the useful or the important ones from the multitude of the arriving messages. These and similar problems are frequently discussed in the case of groups of agents collaborating through networks and formulating and using recommendations for filtering information ([1]).

The solutions of these problems and the answers to these questions suppose having an elaborated semantic background, but to develop suitable and convenient software tools for supporting effective information filtering, also syntactic aspects have to be carefully studied.

A project, titled " Networks of Language Processors" started last year at the Research Group on Modelling Multi-Agent Systems, at the Computer and Automation Research Institute of the Hungarian Academy of Sciences, with the aim of describing at the pure syntactic level characteristics of the behaviour of agents and agent communities using a network for cooperation and communication and to offer tools for designing languages supporting collaborative text processing via networks.

The research is mainly based on tools of formal languages, a traditional area of theoretical computer science, and it is a continuation of investigations that have been done for years by an international team in a recent field of formal language theory called (parallel communicating) grammar systems ([9], [3]).

The developed framework is called networks of language processors (NLP systems). This collective term originally has been introduced as a formal language theoretic framework for describing

[*]Research supported by Hungarian Scientific Research Fund "OTKA" No. T 017 105

symbolic processing in highly (massively) parallel and distributed architectures ([6]). The model was strongly motivated by some known models and paradigms ([7]),[8],[11],[10]). Arguments for formulating such a concept were, among other things, the claim to provide reliable language theoretic support for networked computing, for social networks, for describing the behaviour of mainly locally connected processor arrays, and understanding the nature of massively parallel and distributed architectures, including ones with biological or other nature-motivated background.

NLP systems capture properties of some related notions from formal language theory: the test tube systems ([4]), language theoretic constructs for distributed architectures from DNA computing, parallel communicating grammar systems ([3]), models motivated by distributed and decentralized problem solving systems, grammar systems with WAVE-like communication, providing grammatical models of the so-called Logic Flow paradigm ([5]).

2 Networks of language processors

In the following we briefly describe the main characteristics of the framework. For further details and information the reader is referred to [6] and [2].

A network of language processors (an NLP system, for short) consists of several language determining devices (or mechanisms computing multisets of strings), called language processors. These form the components of the system.

Each language processor represents an agent which processes textual information and cooperates with the other ones by communicating information pieces. Every language processor is located at some node of a virtual graph (a network), moreover, there is no more than one language processor at each node.

The language processors of the NLP system operate on strings (on sets of strings or multisets of strings) by performing rewriting steps and communication steps, usually alternately. The strings can represent data and/or programs (the latter correspond to language theoretic operations in coded form, or to sets of rewriting rules); both kinds of them can be rewritten and communicated. The same string can be interpreted at different components in different manners: it can play the role of a piece of data at some component and that of a rewriting rule at some another one. Thus, the agents can modify the information they have available and they can communicate it to each other. This information can be textual data (strings representing data) or it can be some rule of information handling (program, operation code).

During the functioning of the system new agents can join the network and agents are allowed to leave the agents' community. This leads to a flexible, self-organizing topology of the network. Thus, creation of new components and deletion of some existing ones are allowed, which can be done both as a result of a rewriting step and/or a communication step.

The NLP system is functioning by changing its states (if the rewriting rule sets can be modified, then we use the term "configuration" instead of the term "state"). At any moment of time, the state of the network is described by the sets of string (multisets of strings) present at that moment at the components. Thus, at any moment of time the agents' community is represented by the collection of strings the agents have available at that moment.

At the beginning of functioning, each component of the NLP system is initialized by a language processor (a (finite) set of rewriting rules and the way of its application: for example, a production set of a grammar) and a finite set of initial strings, the axioms. These, together, form the initial state (or the initial configuration) of the system.

The change of the state of the NLP system can take place either by a rewriting step or by a communication step. By a rewriting step, some strings present at some component are rewritten according to the rewriting rule set and rewriting mode of the component (by the metarules in the case of changing the rewriting rules).

By a communication step, some strings (or copies of some strings) which are present at some component and satisfy some condition are communicated to one or more components. The target components are determined by a neighbourhood relation: each language processor is allowed to try to transmit strings only to its neighbours.

Thus, communication is realized through (mainly) local interactions among the components. (Clearly, in some very special cases, the neighbourhood relation makes a broadcast possible.)

The language processors in the network can work either in synchronized or in asynchronous manner.

During the functioning of the NLP system, the communication structure can dynamically vary or it can remain unchanged. The communicated strings are processed by the components which receive them. This can take place in various manners: for example, the arriving strings join the available string community of the component or they are concatenated to some of the strings present at the component.

The conditions for communication can be defined in several ways. One of the most important variants is that one where context conditions are imposed on the strings to check whether the current string can be communicated or not (for example, the existence of some kind of substrings of the string is tested). This, often, is given in the form of *filter* or *selector languages*, that should contain as an element the string to be sent or received. The components can have both an input and an output (entrance and exit) filter. Thus, each agent controls the information flow by using some selector mechanism in order to distinguish useful or important information from the arriving or sent messages. This is similar to what takes place in email systems: some messages have priorities to the other ones both in sending and receiving/reading.

Some components (agents) can have the same input and/or output filter: they form a *team with collective filtering.* These components correspond to a group of agents with the same interest or with the same taste in information selection. The joint filter can be considered as a *recommendation* of the group for its members to select from the information pieces.

The filters either can remain fixed during the functioning of the network or they can dynamically change. In the latter case the team members - depending on the information they have available at some moment of time - change the context conditions representing the input/output selector languages. Thus, at some moment the team of the agents can recommend new rules of information selecting, i.e. they *collaborate in determining filtering conditions.*

We should note that not only the filters but the teams can change during the functioning of the NLP systems: agents are allowed to migrate among the groups or they are allowed to join more than one team. This often takes place in real life: people modify their interest or, simply, to obtain some necessary information they join some new group of interest.

Variants of communication protocols lead to a wide variety of classifications of networks of language processors. If each rewriting step is followed by a communication step, then we speak of NLP systems with language processors communicating by command. If the rewriting at the component continues until a previously prescribed state (a state with a request for communication) is obtained and the communication step takes place afterwards, then we speak of networks of language processors communicating by request.

The above general model offers a language theoretic framework for modelling self-organizing, adaptive, evolving networks of (computational) agents. It is easy to see that it can be considered as a syntactic model of social networks with collaborative filtering of information or a syntactic approach to distributed /cooperative text processing systems realized on networks. The model, because of its general set-up, captures and can be extended to capture features of several extensively and intensively studied variants of networks: as a future example, the dynamic activity pattern of restricted features of the Internet could also be modelled in this way.

NLP systems are both computational and language identifying devices. Both their computational power and computational complexity and their language theoretic properties, including descriptive and size complexity, are of interest. (Languages can be associated with networks of language processors in various manners: for example, we distinguish a master component and take, as the corresponding language, any string that appears at this component during the computation.) In addition, since during their functioning NLP systems determine dynamically changing string multitudes, complexities concerning spatiotemporal dynamics of the emerging string collections are of particular interest. Studying, for example, the occurrence of waves or overloaded situations at the nodes, in the case of string multitudes of networks of language processors can lead to a deeper insight into the nature of distributed and parallel symbol processing and can help in understanding and modelling emerging phenomena in the case of networks like Internet.

3 A formal model

To illustrate the informal framework, we present the formal definition of a variant, called a *network of parallel language processors with teams with collective filtering* (a $TNLP\_F0L$ system, for short). The notion was formulated by some modifications of the notion of a network of parallel language processors ([6]).

In this case the language processors at the components are F0L systems, so-called interactionless Lindenmayer systems with a finite set of axioms, which are, roughly speaking, context-free grammars with a totally parallel way of derivation. (Originally, these systems were introduced for modelling developmental systems in terms of formal grammars, motivated by theoretical biology. The reader can find detailed information on Lindenmayer systems in [9].)

We assume that the reader is familiar with the basics of formal language theory. We list here only some notions which are necessary to follow the ideas of the formal contructions; for more details confer to [9].

For an alphabet V, V^+ denotes the set of all nonempty strings (words) over V. The empty string is denoted by λ, V^* stands for $V^+ \cup \{\lambda\}$. A language L is a subset of V^*.

An $F0L$ system is a triple $H = (V, P, F)$, where V is an alphabet, $F \subset V^*$, is a finite set of axioms, and P is a finite set of productions (rules) of the form $a \to v$, where $a \in V$ and $v \in V^*$. Moreover, production set P is complete: for every $a \in V$ there is a rule of the form $a \to v$, $v \in V^*$ in P. If F consists of exactly one string, then we speak of an $0L$ system. The direct derivation relation in an $F0L$ system $H = (V, P, F)$ is defined as follows: for $x, y \in V^*$ we write $x \Longrightarrow_P y$ if $x = a_1 \dots a_n$, $y = z_1 z_2 \dots z_n$, $a_i \in V$, $z_i \in V^*$, $1 \le i \le n$, and $a_i \to z_i \in P$.

In the following we define the *network of parallel language processors with teams with collective filtering*. We use some simplifications with respect to the general model: the number of the components, the rewriting rule sets of the language processors, the filters and the teams remain unchanged during the functioning of the system, the processors work in synchronized manner and the components check the strings to be communicated by using context conditions. Moreover, each agent is member of exactly one team.

Definition 3.1

A *network of parallel language processors with teams with collective filtering* of degree n, $n \ge 1$, (a $TNLP\_F0L$ system, for short) is a construct

$$, = (V, (\rho_1, \sigma_1, t_1), \dots, (\rho_n, \sigma_n, t_n), R),$$

where

- V is an alphabet (the alphabet of the system),
- ρ_i and σ_i, $1 \le i \le n$, are context conditions over V^* (computable mappings from V^* to $\{\underline{true}, \underline{false}\}$), called the *exit filter* and the *entrance filter* recommended by the i-th team to the members of the team, respectively,
- $t_i = (c_{i,1}, \dots, c_{i,r_i})$, $1 \le i \le n$, $r_i \ge 1$, called a team of components of the system (the i-th team), where
- $c_{i,j} = (P_{i,j}, F_{i,j})$, $1 \le i \le n$, $1 \le j \le r_i$, called a component of the network, the (i, j)-th component, where
- $P_{i,j}$ is a finite set of $F0L$ rules over V, the production set of the (i, j)-th component and
- $F_{i,j} \subset V^*$ is a finite set, the set of axioms of the (i, j)-th component, $1 \le i \le n$, $1 \le j \le r_i$,
- $R \subseteq \Pi \times \Pi$, where $\Pi = \{c_{1,1}, \dots, c_{1,r_1}, \dots, c_{n,1}, \dots, c_{n,r_n}\}$, called the neighbourhood relation of the components of , .

The components represent agents, which by using their sets of rewriting rules can update the textual information they have. Moreover, they form groups (teams), members of which have the same filtering conditions for selecting the information to be communicated and received.

The $TNLP$ system is functioning by changing its states.

By a *state* of a $TNLP\_F0L$ system , $= (V, t_1, \dots, t_n, R)$, $n \geq 1$, we mean a tuple $s = (L_{1,1}, \dots, L_{1,r_1}, \dots, L_{n,1}, \dots, L_{n,r_n})$, where $L_{i,j} \subseteq V^*$, $1 \leq i \leq n$, $1 \leq j \leq r_i$.

L_{i_j} is called the state of the (i, j)-th component and it represents the set of strings which are present at component (i, j) at that moment.

$s_0 = (F_{1,1}, \dots, F_{1,r_1}, \dots, F_{n,1}, \dots, F_{n,r_n})$ is said to be the *initial state* of the system.

A state can change either by a *rewriting step* or by a *communication step.* When a rewriting step takes place, then every component derives from each available string a new one, by applying its productions in the $F0L$ manner. Thus, in this case the number of strings available at the components does not change: each agent has the same number of strings as it had before the rewriting step.

At a communication step, each component (i, j) receives a copy of all strings that are present at some of its neighbourhood components, say, component (k, l) and are able to pass the exit filter of component (k, l) - this is the exit filter recommended to use by team k - and the entrance filter of component (i, j) - the entrance filter recommended by team i for receiving messages. (These strings satisfy context conditions ρ_k and σ_i).

Definition 3.2

Let , $= (V, t_1, \dots, t_n, R)$, $n \geq 1$, be a $TNLP\_F0L$ system.

Let $s_1 = (L_{1,1}, \dots, L_{1,r_1}, \dots, L_{n,1}, \dots, L_{n,r_n})$, and $s_2 = (L'_{1,1}, \dots, L'_{1,r_1}, \dots, L'_{n,1}, \dots, L'_{n,r_n})$ be two states of , . We say that

- s_1 directly changes for s_2 by a *rewriting step*, written as
$$(L_{1,1}, \dots, L_{1,r_1}, \dots, L_{n,1}, \dots, L_{n,r_n}) \Longrightarrow (L'_{1,1}, \dots, L'_{1,r_1}, \dots, L'_{n,1}, \dots, L'_{n,r_n})$$
if $L'_{i,j}$ is the set of words obtained by performing a derivation step on each element of $L_{i,j}$ by production set $P_{i,j}$ in the $F0L$ manner, $1 \leq i \leq n, 1 \leq j \leq r_i$,

- s_1 directly changes for s_2 by a *communication step* in , , written as
$$(L_{1,1}, \dots, L_{1,r_1}, \dots, L_{n,1}, \dots, L_{n,r_n}) \vdash (L'_{1,1}, \dots, L'_{1,r_1}, \dots, L'_{n,1}, \dots, L'_{n,r_n})$$
if for every i, $1 \leq i \leq n$, and j, $1 \leq j \leq r_i$,
$L'_{i,j} = L_{i,j} \cup \{v \mid v \in L_{k,l},\ \rho_k(v) = \underline{true}$ and $\sigma_i(v) = \underline{true}, 1 \leq k \leq n, 1 \leq l \leq r_k, (k,l) \neq (i,j), (c_{i,j}, c_{k,l}) \in R\}$.

Notice that according to the above definition an agent in team i is allowed to receive messages from another agent of the same team.

A sequence of subsequent states determines a computation in , .

Let , $= (V, t_1, \dots, t_n, R)$, $n \geq 1$, be a $TNLP\_F0L$ system. By a *computation* C in , we mean a sequence of states $s_0, s_1, \dots$, where

- $s_i \Longrightarrow s_{i+1}$ if $i = 2j$, $j \geq 0$, and
- $s_i \vdash s_{i+1}$ if $i = 2j + 1$, $j \geq 0$.

Let , $= (V, t_1, \dots, t_n, R)$, be a $TNLP\_F0L$ system.

The *language* $L($, $)$ determined by , is

$L(,) = \{w \in L_1^{(s)} \mid (F_{1,1}, \dots, F_{n,r_n}) = (L_{1,1}^{(0)}, \dots, L_{n,r_n}^{(0)}) \Longrightarrow (L_{1,1}^{(1)}, \dots, L_{n,r_n}^{(1)}) \vdash (L_{1,1}^{(2)}, \dots, L_{n,r_n}^{(2)}) \Longrightarrow \dots \Longrightarrow (L_{1,1}^{(s)}, \dots, L_{n,r_n}^{(s)}),\ s \geq 1\}$.

4 On the power of TNLP systems

Networks of language processors are language determining (computational) devices, therefore the question how large language classes (how complicated string communities) can be computed by their particular variants is one of the most important questions. Especially interesting are those NLP systems which are of considerable computational power and at the same time with extremely simple presentation. Simplicity in this case means, among other things, restricted size parameters of the network (a small number of components), poor power of the language theoretic operation represented by the language processor (restricted capabilities of the agents), homogenity of the components, simple communication protocol and simple (regular, subregular) filter languages.

Networks of language processors with $F0L$ systems as components and with filter languages defined by regular context conditions form a computational device equally powerful to the Turing machine ([6]). (To pass a regular filter, the string have to be an element of the regular language identifying the filter.) Morevover, it can be shown that in the case of regular filters a bounded number of components is sufficient to reach computational completeness. The same results can be derived for TNLP systems. Thus, TNLP systems with parallel language processors even with very simple presentation and with relatively simple filtering are able to process very complicated string collections.

Classes of languages determined by several kinds of networks of language processors based on different language theoretic operations have been studied in detail: it was shown, for example, that networks of language processors with regular filters and with context-free grammars as language processors or with language processors based on language theoretic operations simulating the recombinant behaviour of DNA strands or with operations corresponding to point mutations (splicing, cutting and recombination, insertion, deletion, replacement, etc.) provide universal computing devices. (For an overview on the area the interested reader is referred to [2].)

5 String collections of TNLP systems

Networks of language processors are devices not only for describing the dynamics of languages at the components but they also provide tools for characterizing multisets of strings. Properties of these string collections are of particular importance in those cases when not only the information piece itself (for example, the arriving message), but the number of its available copies is of interest. Since the notions related to these networks of string multiset processors (NMP systems) are isomorphic to the notions concerning NLP systems, we omit the explicit definitions. We only note that in this case the computing devices located at the components operate on such collections of strings where the strings are allowed to have multiple (a finite number of) occurrences of the same copy.

In [6] it was shown that the growth of the number of strings present during the computation at the components of an NMP system which has random context filters and deterministic F0L systems as components can be described by the growth function of a D0L system. (A D0L system is an 0L system with exactly one production for each letter a of the alphabet at the left-hand side. A random context filter checks the string according to the presence/absence of some symbols. The growth function of a D0L system orders to each natural number n the length of the word generated by the system at the n-th step of the derivation.)

The proof is based on the following simple considerations: since $D0L$ systems define homomorphisms, therefore if we know how many strings with a fixed alphabet are present at some component, then we are able to give the number of strings with the same alphabet obtained after performing a rewriting step at the component. Moreover, because at communication steps we check the presence/absence of some symbols in the strings, we are able to decide whether a string with a fixed alphabet can pass a filter or not. Thus, at any state of the computation we can represent the multiset of strings at some component by the multiset of their alphabets, and, we can construct a $D0L$ system such that the multiset of the letters of the word of the $D0L$ system at step t is equal to the multiset of the alphabets of the strings present at some component (at the components) at a corresponding step of computation in the network.

Using this proof technique, the same result can be given in the case of TNMP systems with deterministic F0L components and random context filters. Moreover, we note that not only the growth of the whole string community, but also the growth of the number of strings at the individual components and teams can be calculated. By the theory of D0L growth functions we can derive several interesting properties of the emerging string collections at the TNMP systems. We know, for example, that the growth of the string population (at some team or at some individual component) is either polynomially bounded or exponential and this is a decidable property.

6 Final remarks

In this paper we briefly discussed a general framework which provides language theoretic approach for describing the behaviour of agents and agent communities which use networks for cooperation and communication. We hope that the theoretical model can help in developing tools for designing languages supporting collaborative text processing via networks.

References

[1] Communication of the ACM, March 1997, Vol. 40., No. 3. Special issue: Recommender Systems. Linking users by similar interest.

[2] E. Csuhaj-Varjú, Networks of Language Processors. EATCS Bulletin, 63 (1997), 120-134.

[3] E. Csuhaj-Varjú, J. Dassow, J. Kelemen and Gh. Păun, Grammar Systems: a Grammatical Approach to Distribution and Cooperation. Gordon and Breach Science Publisher, London, 1994.

[4] E. Csuhaj-Varjú, L. Kari and Gh. Păun, Test Tube Distributed Systems Based on Splicing. Computers and Artif. Intelligence 15(2-3) (1996), 211-232.

[5] E. Csuhaj-Varjú, J. Kelemen and Gh. Păun, Grammar Systems with WAVE-like Communication. Computers and Artif. Intelligence 15 (5) (1996), 419-436.

[6] E. Csuhaj-Varjú and A. Salomaa, Networks of Parallel Language Processors. In: New Trends in Formal Languages. Control, Cooperation and Combinatorics, (Gh. Păun, A. Salomaa, eds.), LNCS 1218, Springer Verlag, Berlin-Heidelberg-New York, 1997, 299-318.

[7] L. Errico and C. Jesshope, Towards a new architecture for symbolic processing. In: Proc. Conf. Artificial Intelligence and Information-Control Systems of Robots' 94, (I. Plander, ed.), World Scientific, Singapore, 1994, 31-40.

[8] S. E. Fahlman, G. E. Hinton, T. J. Seijnowski, Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. In: Proc. AAAI- Natl. Conf. on AI., William Kaufman, Los Altos, 1983, 109-113.

[9] Handbook of Formal Languages. Vol. I-II-III. (G. Rozenberg, A. Salomaa, eds.), Springer Verlag, Berlin-Heidelberg-New York, 1997.

[10] C. Hewitt, Viewing Control Structures as Patterns of Passing Messages. J. of Artificial Intelligence 8 (1977), 323-364.

[11] W. D. Hillis, The Connection Machine, MIT Press, Cambridge, 1985.