ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **12**, Number 2, 2009, 125–137

Using Finite-State Methods for Getting Infinite Languages: A Preview

Gemma BEL-ENGUIX, Maria Dolores JIMÉNEZ-LÓPEZ, Carlos MARTÍN-VIDE

Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco, 1, 43005 Tarragona, Spain
E-mail: {gemma.bel,mariadolores.jimenez,carlos.martin}@urv.cat

Abstract. In this paper, a new/alternative approach to language representation is presented. In our model, languages are formed by interactions of a finite number of grammars each of them generating just a finite language. We consider languages as infinite objects which *emerge* from interactions of 'smaller' finite language generators. The approach we introduce here can build a bridge between two opposite views in present-day linguistics –one that regards natural languages as infinite objects, and the second one that regards them as finite objects. The basic notion we used here is called a *colony*. Colonies as well-formalized language generating devices were proposed in [33]. *Components* of a colony are *regular grammars* generating finite languages and operating on a shared string of symbols. The aim of this contribution is to connect both colonies and the philosophy behind them with the opposition between two conceptions of natural language: the work in *theoretical linguistics*, where many scholars agree that the set of sentences in a human language must be denumerably infinite, and the work in *corpus-based computational linguistics*, where the focus is often on language as a finite set. We intend to show that it is possible to formally reconcile these two so apparently different conceptions of natural language.

1. Introduction

In the traditional theory of formal languages, a language is considered as a possible infinite object generated by a finite number of rules of the corresponding formal

grammar(s). In this contribution, an alternative approach to language representation is presented, where languages are formed by interactions of a finite number of grammars each of them generating just a finite language.

By considering languages as infinite objects that emerge from interactions of 'smaller' finite languages generators, the approach introduced here can build a bridge between two opposite views in present-day linguistics –one that regards natural languages as *infinite objects*, and another that regards them as *finite objects*. This opposition between two conceptions of natural language roughly coincides, respectively, with the work in *theoretical linguistics*, where many scholars agree that the set of sentences in a human language must be denumerably infinite, and the work in *corpus-based computational linguistics*, where the focus is often on language as a finite set. We intend to show that it is possible to formally reconcile these two so apparently different conceptions of natural language and that they are not incompatible.

The basic notion used for the formal definition of the above-mentioned type of generative device is called a *colony*. Colonies as well-formalized language generating devices have been proposed in [33], and developed during the nineties in several directions in a lot of papers, e.g. [35], [48], [59], [44], [34], [58], [14].

In the last decade, computational models have become mostly bio-inspired. In the same way, the basic concept of colony, that is taken first from nature, has been developed by means of several bio-inspired computing theories, giving rise to membrane systems [49], tissue P systems [45] or NEPs [9]. Despite the differences, the main idea of colonies remains in these models: *interaction, collaboration, emergence*. The most relevant contribution of bio-inspired models to the basic formalization seems to be the concept of evolution in the configuration and definition of the components of the system during the computation.

This paper takes as starting point the primary theory of colonies. This model provides a simple and consistent basis for our purpose: to approach the issue of the interplay between language and computation. The aim of this contribution is to connect both colonies and the philosophy behind them with the above-mentioned linguistic problems.

2. Natural Languages: Finite or Infinite?

The question whether the grammatical sentences of natural languages form regular, context-free, context-sensitive or recursively enumerable sets has been subject to much discussion since it was posed by Chomsky in 1957. There seems to be little agreement among linguists concerned with the position of natural languages.

The current debate on the complexity of natural language has focused on finding adequate models of certain phenomena (embedding, reduplication, cross-serial dependencies) and investigating their formal properties. This approach makes it possible to distinguish between constructions requiring at least the expressive power of a context-free grammar and those which can be modelled by lower-level methods.

Although Chomsky attempted to demonstrate the inadequacy of finite-state devices for natural languages, several linguists have advocated their use.

We can briefly review these two positions by formulating the following question: *Are natural languages finite or infinite?* In order to answer such a question, we have to start by establishing a distinction between:

1. those that think that natural languages are finite (vs. those that believe that they are infinite.)
2. those that think that finite-state devices are adequate to characterize natural languages.

Notice that the above positions constitute two ways of understanding the idea of finiteness, this is: *natural languages as finite objects vs. adequacy of finite-state devices for natural languages.* In the remain part of this section we will review these two positions.

2.1. Finite *versus* Infinite

According to Savitch [54], there are clear senses in which natural languages are finite. Among them, the author enumerates the following ones:

1. it is generally assumed that human brains and hence linguistic processing ability is finite;
2. the available data does not in any obvious way support the hypothesis that natural language is infinite, rather than simply a large finite set;
3. at any point in time, an individual, or the entire community of human beings, will have experienced only a finite number of sentences. At no point in history will scientists be able to point to any actual pool explicitly containing an infinite number of sentences.

Despite those apparently irrefutable facts, linguists treat natural languages as infinite sets, at least when studying syntax. In this respect we can refer to Chomsky's words:

> *"In general it is assumed that natural languages are infinite in order to simplify their description. If a grammar does not have recursive devices, it will be prohibitively complex. If it has recursive devices of any type, it will produce an infinite number of sentences"* [10]

> *"Structurally, human language is a system with recursive structure-dependent rules, operating on sequences organized in a hierarchy of phrases to generate a countable infinity of sentences"* [11]

2.2. Recognizable by Finite-State Machines

In what regards to the mechanism used, we can see that, in general, linguists agree that syntax of a natural language cannot be described by a finite-state device or even

a context-free grammar. The frequently postulated need for very powerful tools is motivated by different kinds of recursion and embedding. Indeed, in order to generate an appropriate structure for those facts a formalism having the generative capacity of context-free grammar is required. Nevertheless, it seems that there are many subsets of natural language that can be correctly described by very simple means, this is, that can be handled by less powerful devices. Examples of those sublanguages could be names and titles, addresses, prices, dates, etc. For some of those kinds of expressions, a finite-state grammar may be more appropriate and easier to construct than an ordinary phrase structure grammar.

Taking into account the above-mentioned facts, finite-state devices, in one form or another, have been used for the description of natural language since the early 1950s. But, after Chomsky's condemnation of the adequacy of finite-state devices for describing sentence structures, they virtually disappear from theoretical linguistics throughout the seventies and eighties, and they just reappear on the scene again thanks to C. Douglas Jonhson's demonstration that it only takes a finite-state machine to model a phonological rule –instead of the used context-sensitive rewrite rules–, and to Kaplan and Kay paper in which the same conclusion was reached [19].

The argument that makes that many linguists and computational linguists came to believe that finite-state devices were of little or no interest was the following Chomsky's assertion:

> *"Any attempt to construct a finite-state grammar for English finds difficulties and complications. [...] Therefore, it seems quite clear that no theory for the linguistic structure based exclusively on Markov models, or models of similar nature, will be able of explaining or giving information of English speakers ability of producing and understanding new sentences. [...] A finite-state grammar is the simplest type of grammar that with a finite number of mechanisms can generate an infinite number of sentences. A so limited linguistic theory is not adequate; we are forced, therefore, to look for a more powerful type of grammar."* [10]

It may be true that complete and accurate natural language syntactic and semantic dependencies lie beyond the power of finite-state description, but work in the last years has identified a number of important problems for which very efficient and effective finite-state solutions can be found. It seems that finite-state descriptions can provide an approximation to the proper grammatical description of a language, an approximation that is often good enough for *practical* purposes.

Following the idea of the utility of finite-state devices in natural language –and taking into account the wide use of those devices in many areas of computer science– there has been a resurgence of the use of finite-state techniques in many aspects of computational linguistics, from the construction of lexical analyzers and the compilation of morphological and phonological rules to speech processing. The use of finite-state devices in many areas of computational linguistics can be justified by linguistic and computational arguments. Linguistically finite-state devices are convenient since they allow to describe easily most of the relevant local phenomena encountered in the empirical study of language. From the computational point of view, the use of

finite-state devices is mainly motivated by considerations of time and space efficiency. Among the applications of finite-state devices in natural language processing, we can refer to the following ones [52, 64]:

1. *Dictionary Encoding.* There has been a growing interest in using finite-state devices for storing and accessing natural languages dictionaries [2, 42].

2. *Morphology.* Morphological analysers for various languages including Dutch, English, French, German, Italian, Portuguese, Spanish, etc. have been developed. It seems that morphological analysis is inherently finite-state in character and finite-state solutions for this problem are complete and efficient [5, 6, 28, 7].

3. *Part-of-Speech Disambiguation (Tagging).* The general purpose of a part-of-speech tagger is to associate each word in a text with its morphosyntactic category. The process consists in three steps: (1) *tokenization*: break a text into tokens; (2) *lexical lookup*: provide all potential tags of each token; (3) *disambiguation*: assign to each token a single tag [57, 51].

4. *Parsing.* Finite-state parsing is an extension of finite-state devices to the level of phrases and sentences [1, 15, 41, 50].

5. *Information Extraction.* This problem requires documents and passages be identified that are likely to contain information relevant to a given user's needs. Full-blown syntactic and semantic analyses of documents would certainly help to solve this problem. But such analyses may provide much more information than this task actually requires. A finite-state solution has been constructed that is extremely efficient compared to more powerful but more complex extraction systems and also has very favourable recall and precision scores [39, 40, 3].

So, in general, it seems that there are regular subsets of natural language for which finite-state description is not only feasible but more appropriate and easier to construct than the equivalent phrase-structure grammar. If the language to be described is in fact regular, there many be a significant advantage in describing it by means of a regular grammar instead of using a more powerful grammar formalism.

Summing up, it seems that although regular grammars can cover only limited subsets of a natural language, there could be a practical advantage in describing such sublanguages by means of regular expressions rather than by some more powerful formalisms.

In this section, we have reviewed two different positions about the finiteness or infiniteness of natural languages:

1. Firstly, we have referred to two ways of *regarding* natural languages: as finite or infinite objects.

2. Secondly, we have presented two ideas of *how to describe* natural languages: on the one hand, we have seen that there are scholars that defend that finite-state devices could be adequate for describing natural languages; on the other hand, we find those that deny the adequacy of such finite-state devices and postulate the necessity of more powerful models in order to well describe natural language.

If we assume now that natural languages are infinite objects –this seems to be the more accepted idea, in spite of the reasons adduced on the contrary–, the question is: Can different ways of describing natural language –with finite-state devices or more powerful mechanisms– be considered two *opposite and irreconcilable* views in present-day linguistics? We think that it is not possible to regard those two positions as a hard debate about the *finiteness* or *infiniteness* of natural languages. The point is not that some authors think that natural languages are infinite while others think that they are finite. Rather than a debate, what we have here are just two different ways of handling natural languages: either by finite-state devices –the way chosen by those who work on computational linguistics– or by more powerful devices –the way chosen by those who work on theoretical linguistics. Therefore, we cannot talk about two irreconcilable views, because these two ways of describing natural language corresponds to two ways of doing linguistics, to two ways of approaching natural languages –either from a *theoretical* or from a *practical* point of view.

3. Colonies: A Finite-State Device for the Emergence of Infinite Languages

The above section has shown that we can afford natural languages either from a *theoretical* point of view or from a *practical* point of view. Depending on our interest, we will defend the adequacy or not adequacy of finite-state devices to model natural language. Here, we propose *colonies* as a model that allow us to defend the infiniteness of natural languages while treating them with finite-state devices. Therefore, colonies can be regarded as a model that may formally reconcile these two so apparently different conceptions of natural language, showing that they are not incompatible, but that they can even be simultaneously defended in the same model.

3.1. Colonies: Formal Definition

Colonies as well-formalized language generating devices have been proposed in [33], and developed during the nineties in several directions in many papers.

Colonies can be thought of as grammatical models of multi-agent systems motivated by Brooks' subsumption architectures [8]. They describe language classes in terms of behaviour of collections of very simple, purely reactive, situated agents with emergent behaviour. Roughly, a colony consists of a finite number of simple modules (regular grammars) which generate finite languages and operate on a shared string of symbols –the *environment* of the colony– without any explicitly predefined strategy of cooperation. Each component has its own reactive behaviour which consists in: 1) sensing some aspects of the *context* and 2) performing elementary tasks in it in order to achieve some local changes. The environment is quite passive, its state changes only as result of acts agents perform on it. Because of the lack of any predefined strategy of cooperation, each component participates in the rewriting of current strings whenever it can participate in it. The behaviour of a colony –this is, the language– is defined as the set of all the strings which can be generated by the colony from a given

starting string. In what follows we introduce the formal definition of a colony. For formal results and new variants on this area we refer to [29, 30, 31, 32, 12, 43, 44, 4].

Definition 1. A *colony* C is a 3-tuple:

$$C = (R, V, T)$$

where

(i) $R = \{R_i | 1 \leq i \leq n\}$ is a finite set of regular grammars $R_i = (N_i, T_i, P_i, S_i)$ producing finite languages $L(R_i) = F_i$ for each i. R_i will be referred to as a component of C.

(ii) V $= \bigcup_{i=1}^{n} (T_i \cup N_i)$ is the alphabet of the colony.

(iii) T $\subseteq V$ is the terminal alphabet of the colony.

We note that terminal symbols of one grammar can occur as nonterminals symbols of another grammar.

Components $R_i \in R$ $(1 \leq i \leq n)$ of a colony C are *regular grammars* (as known in the traditional theory of formal grammars and languages) generating finite languages and operating on a shared string of symbols –the *environment* of the colony– without any explicitly predefined strategy of cooperation of the components.

An *environment* of a colony is formed by strings of symbols from V. Strings are modified only by sequential activities of components of a colony. Because of the lack of any predefined strategy of cooperation between components, each component participates in the rewriting of the current string whenever its start symbol is present in the actual string. Conflicts are solved non-deterministically, as it is usual in the classical theory of formal grammars.

The activity of components in a colony is realized by string transformation on a common tape. Elementary changes of strings are determined by a basic derivation step:

Definition 2. For $x, y \in V^*$ we define $x \Longrightarrow y$ iff

$$x = x_1 S_i x_2, \quad y = x_1 z x_2, \text{ where } z \in F_i \text{ for some } i, 1 \leq i \leq n.$$

The behaviour of a colony is defined as the set of all strings which can be generated by the colony from a given starting string. A terminal symbol of one component can occur as a non-terminal symbol of another one, so that the possibility of cooperation of components of the colony allows to generate substantially more than finite languages. The global behaviour of the whole colony emerges from the strictly individual behaviours of components.

Definition 3. The language determined by a colony C starting with the word $w_0 \in V^*$ is given by:

$$L(C, w_0) = \{v | w_o \Longrightarrow^* v, v \in T^*\}.$$

The *language* $L_C \subseteq T^*$ generated by a colony C is a subset of all strings over the terminal alphabet T which are generated by the components from R of the colony C starting from a given string.

A graphical view of a colony is presented in Fig. 1.

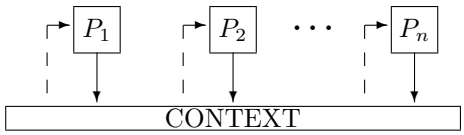

Fig. 1. A colony.

3.2. Colonies as a Model for Natural Language

A colony, as proposed originally, captures some formal aspects of systems of finite number of autonomous components capable to perform very simple reactive computing tasks each. The behaviour of the colony really emerges from interactions of its components with their symbolic environment and can considerably surpass the individual behaviours of its components.

Colonies offer a formal framework for description and study of the emergence of complicated behaviours from simple ones of their purely reactive components.

With colonies, we can generate natural language by the interaction of a finite number of finite-state devices that generate finite languages. If we use colonies we do not need to consider a priori that natural languages are infinite, we just need to construct different finite-state grammars each generating a finite language. The result of such interaction will be what we are waiting for: *infiniteness*. Notice that in order to get *infinite* languages we use *finite* means. The infinite *emerges*.

In computational linguistics, finite-state devices have been used for phonology, morphology, lexicon and syntax of natural languages. The advantages of approaching such parts of natural languages with finite-state devices have been demonstrated. So, from a practical point of view, finite-state descriptions of natural languages have revealed as adequate. The problems of using such devices arise when we consider natural languages as a whole, pointing our attention on the complexity of such an object and on its infinite nature. Here is where colonies can enter to reconcile these two positions: for each part of a natural language, we define a *finite-state grammar* that generates a finite language –defending, as in *computational linguistics*, the adequacy of such mechanisms. The result of the interaction of those finite-state grammars will be *infinite* -as defended in *theoretical linguistics*.

Specifically, what we propose is to define finite-state devices for: morphology, phonology, syntax, lexicon and any other required module in language. If we put those finite-state devices in a colony, they will interact in order to generate natural language. The result of the interaction will be, not a regular language, but a language that is context-free or more.

In [33] a formal theoretic result is proved according to which an arbitrary context-free formal language can be generated by a corresponding colony, and an arbitrary

formal language generated by a colony is context-free. In other words, the family of languages generated by colonies is identical to the family of all context-free languages. Using this statement, and taking into account the fact that the family of context-free languages is infinite, we can say that by putting finite-state devices in a colony we get, not finite languages, but infinite languages thanks to the interaction in the colony. The idea is clear: *if we have finite-states devices that interact in a colony, the infinite emerges.* In this way we put together the apparent different ideas defended in Computational and Theoretical Linguistics.

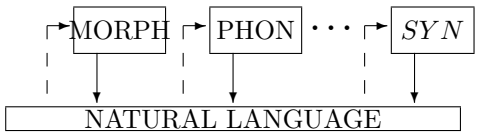

Fig. 2. A colony for Natural Language.

4. Final Remarks

Natural languages can be described as a number of modules that interact in a nonsimple way. Understanding and generation needs cooperative phonological, morphological, lexical, syntactic, pragmatic, semantic... modules. In general, formal and computational approaches to natural language demand distributed models in order to explain the complexity of linguistic structures as the result of the interaction of a number of independent but cooperative modules. Colonies offer a modular theory where the various dimensions of linguistic representation are arranged in a distributed framework and where the language of the system is the result of the interaction of those independent cooperative modules.

The main advantage of colonies is their generative power, the class of languages describable by colonies that make use of strictly regular components is beyond the set describable in terms of individual regular grammars.

According to [63], the attractiveness of finite-state technology for natural language processing stems from four sources:

1. *modularity* of the design, due to the closure properties of regular languages and relations
2. *compact representation* that is achieved through minimization
3. *efficiency*, which is a result of linear recognition time with finite-state devices
4. *reversibility*, resulting from the declarative nature of finite-state devices

In the field of computational linguistics, and because of their mathematical and computational simplicity (in terms of time and space efficiency), regular languages and

finite-state automata are applied to many issues related to natural language processing. However, in theoretical linguistics, we see that one of the most persistent issues has been the one of determining where natural languages are located on 'Chomsky hierarchy', and, in general, the answer has been that natural language must be, at least, context-free. So, from a theoretical point of view, regular languages seem not to be adequate to characterize a natural language.

In this paper, we have proposed colonies as a tool that may allow us to generate infinite languages by only using regular grammars. This formal framework increases the power of regular grammars thanks to interaction. What is important here is the fact that although the generative power of colonies goes beyond the regular family of languages, the derivation process is done in a regular (finite-state) manner. So, colonies may reveal as a device able of conjoining the simplicity of finite-state machines with a stronger generative power able to account for the infiniteness (context-free or more) of natural languages.

Ideas introduced in this paper still seminal, but we think that colonies can formally reconcile the two different conceptions of natural language: the work in *theoretical linguistics*, where many scholars agree that the set of sentences in a human language must be denumerably infinite, and the work in *corpus-based computational linguistics*, where the focus is on language as a finite set.

References

[1] AÏT-MOKHTAR S., CHANOD J.P., *Incremental Finite-State Parsing*, *Proceedings of ANLP'97*, Washington, 1997.

[2] APPEL A.W., JACOBSON G., *The World's fastest scrabble program*, Communications of the ACM, **31(5)**, 1986, pp. 572–578.

[3] APPELT D., HOBBS J., BEAR J., ISRAEL D., TYSON M., *FASTUS: A Finite-State Processor for Information Extraction from Real-World Text*, *International Joint Conference on Artificial Intelligence*, 1993, pp. 1172–1178.

[4] BANÍK I., *Colonies with Position*, Computers and Artificial Intelligence, **15**, 1996, pp. 141–154.

[5] BEESLEY K.R., *Arabic Morphological Analysis on the Internet*, *Proceedings of the International Conference and Exhibition on Multi-lingual Computing (Arabic and English), ICEMCO-98*, 1998.

[6] BEESLEY K.R., KARTTUNEN L., *Finite-State Non-Concatenative Morphotactics*, in Karttunen L., Eisner J. & Theriault A. (eds.), *SIGPHON-2000. Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology*, Luxembourg, 2000, pp. 1–12.

[7] BEESLEY K., KARTTUNEN L., *Finite State Morphology*, CSLI Publications, Standford, 2003.

[8] BROOKS R.A., *Elephants don't play chess*, Robotics and Autonomous Systems, **6**, 1990, pp. 3–15.

[9] CASTELLANOS J., MARTÍN-VIDE C., MITRANA V., SEMPERE J. M., *Networks of evolutionary processors*, Acta Informatica, **39**, 2003, pp. 517–529.

[10] CHOMSKY N., *Syntactic Structures*, Mouton, The Hague, 1957.

[11] CHOMSKY N., *Rules and Representations*, Blackwell, Oxford, 1980.

[12] DASSOW J., KELEMEN J., PĂUN G., *On Parallelism in Colonies*, Cybernetics and Systems, **14**, 1993, pp. 37–49.

[13] EJERHED E., *Finite State Segmentation of Discourse into Clauses*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1996.

[14] GAŠO J., *Unreliable Colonies as Systems of Stochastic Grammars*, Journal of Automata, Languages, and Combinatorics, **5**, 2000.

[15] GREFENSTETTE G., *Light Parsing as Finite-State Filtering*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[16] JOSHI A., *A Parser from Antiquity: An Early Application of Finite State Transducers to Natural Language Parsing*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[17] KAPLAN R.M., *Finite State Technology*, `http://cslu.cse.ogi.edu/HLTsurvey/ch11node8.htlm`

[18] KAPLAN R.M., KAY M., *Finite-State Methods in Natural Language Processing: 1-Motivation*, `http://www.stanford.edu/class/ling138/wk6/sld001.html`

[19] KAPLAN R.M., KAY M., *Regular Models of Phonological Rule Systems*, Computational Linguistics, **20**:3, 1994, pp. 331–378.

[20] KARTTUNEN L., *Directed Replacement*, *Proceedings of the 34th Annual Meeting of the ACL*, Santa Cruz, CA, 1996.

[21] KARTTUNEN L., *Finite-State Constraints*, *Proceedings of the International Conference on Current Issues in Computational Linguistics*, University Saints Malaysia, Penang, Malaysia, 1991, pp. 10–14.

[22] KARTTUNEN L., *Finite-State Lexicon Compiler, Technical Report*, ISTL-NLTT-1993-04-02, Xerox Palo Alto Research Center, California, 1993.

[23] KARTTUNEN L., *Constructing Lexical Transducers*, *Proceedings of the 15th International Conference on Computational Linguistics. Coling 94*, Kyoto, Vol. I, 1994, pp. 406–411.

[24] KARTTUNEN L., *The Replace Operator*, *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics. ACL-95*, Boston, 1995, pp. 16–24.

[25] KARTTUNEN L., *Applications of Finite-State Transducers in Natural-Language Processing*, in Yu S. & Păun G. (eds.), *Implementation and Application of Automata*, LNCS 2088, Springer, Berlin, 2001, pp. 34–46.

[26] KARTTUNEN L., BEESLEY K.R., *Two-Level Rule Compiler*, Technical Report, ISTL-92-2, Xerox Palo Alto Research Center, California, 1992.

[27] KARTTUNEN L., CHANOD J.P., GREFENSTETTE G., SCHILLER A., *Regular Expressions for Language Engineering*, Natural Language Engineering, Cambridge University Press, 1997, pp. 1–24.

[28] KARTTUNEN L., KAPLAN R.M., ZAENEN A., *Two-Level Morphology with Composition*, in *Proceedings of Coling 92. International Conference on Computational Linguistics*, Nantes, Vol. **1**, 1992, pp. 141–148.

[29] KELEMEN J. (ed.), *Grammar Systems and Colonies*, Computer Science Preprint no. CS4-91, Faculty of Mathematics and Physics, Comenius University, Bratislava, 1991.

[30] KELEMEN J. (ed.), *Grammar Systems and Colonies II*, Computer Science Preprint no. CS2-92, Faculty of Mathematics and Physics, Comenius University, Bratislava, 1992.

[31] KELEMEN J. (ed.), *Grammar Systems and Colonies III*, Computer Science Preprint no. CS1-93, Faculty of Mathematics and Physics, Comenius University, Bratislava, 1993.

[32] KELEMEN J., *Colonies – A Theory of Reactive Agents*, in Kelemenová A. (ed.), *Proceedings on the MFCS'98 Satellite Workshop on Grammar Systems*, Silesian University, Faculty of Philosophy and Sciences, Institute of Computer Science, Opava, 1998, pp. 7–38.

[33] KELEMEN J., KELEMENOVÁ A., *A Grammar-Theoretic Treatment of Multiagent Systems*, Cybernetics and Systems, **23**, 1992, pp. 621–633.

[34] KELEMENOVÁ A., *Timing in colonies*, in Păun G. & Salomaa A. (eds.) *Grammatical Models of Multi-Agent Systems*, Gordon and Breach, London, 1999.

[35] KELEMENOVÁ A., CSUHAJ-VARJÚ E., *Languages of Colonies*, Theoretical Computer Science, **134**, 1994, pp. 119–130.

[36] KORNAI A., *Natural Languages and the Chomsky Hierarchy*, in King M. (ed.), *Proceedings of the 2nd European Conference of the Association for Computational Linguistics*, 1985, pp. 1–7.

[37] KORNAI A., *Language Models: Where are the Bottlenecks?*, AISB Quarterly, **88**, 1994, pp. 36–40.

[38] KORNAI A., *Extended Finite State Models of Language*, Natural Language Engineering, **4**, 1996, pp. 287–290.

[39] KOSKENNIEMI K., *Finite-State Morphology and Information Retrieval*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[40] KUSHMERICK N., *Finite-State Approaches to web information extraction*, *Proceedings of the 3rd Summer Convention on Information Extraction*, Springer, Berlin, 2002, pp. 77–91.

[41] LANGENDOEN T., LANGSAM Y., *On the Design of Finite Transducers for Parsing Phrase-Structure Languages*, in Manaster-Ramer A. (ed.), *Mathematics of Language*, John Benjamims, Amsterdam, 1987, pp. 191–235.

[42] LUCCHESI C.L., KOWALTOWSKI T., *Applications of finite automata representing large vocabularies*, Software-Practice and Experience, **23(1)**, 1993, pp. 15–30.

[43] MARTÍN-VIDE C., PĂUN G., *PM-colonies*, Computers and Artificial Intelligence, **17**, 1998, pp. 553–582.

[44] MARTÍN-VIDE C., PĂUN G., *New Topics in Colonies Theory*, Grammars, **1**, 1999, pp. 209–223.

[45] MARTÍN-VIDE C., PĂUN G., PAZOS J., RODRÍGUEZ-PATÓN A. (2002), *Tissue P Systems*, Theoretical Computer Science, **296** (2), 2002, pp. 295–326.

[46] MOHRI M., *Finite State Transducers in Language and Speech Processing*, Computational Linguistics, **vol. 23**, no. 2, 1997, pp. 269–311.

[47] OEHRLE D., *Finite-State Methods, Binding, and Anaphora*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[48] PĂUN G., *On the Generative Power of Colonies*, Kybernetika, **31**, 1995, pp. 83–97.

[49] PĂUN G., *Computing with membranes*, Journal of Computer and Systems Sciences, **61**(1), 2000, pp. 108–143.

[50] ROCHE E., *Finite-State Transducers: Parsing Free and Frozen Sentences*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[51] ROCHE E., SCHABES Y., *Deterministic Part-of-Speech Tagging with Finite State Transducers*, Computational Linguistics, vol. **21**, 1995, pp. 227–253.

[52] ROCHE E., SHABES Y. (Eds.), *Finite-State Language Processing*, MIT Press, Cambridge, 1997.

[53] ROOD C.M., *Efficient Finite-State Approximation of Context Free Grammars*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[54] SAVITCH W., *Why It Might Pay to Assume That Languages Are Infinite*, Annals of Mathematics and Artificial Intelligence, **8** (1–2), 1993, pp. 17–25.

[55] SCHILLER A., *Multilingual Finite-State Noun Phrase Extraction*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[56] SKUT W., *Finite Automata for Processing Word Order*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[57] SKUT W., ULRICH S., HAMMERVOLD K., *A Generic Finite State Compiler for Tagging Rules*, Machine Translation, vol. **18**, Issue 3, 2003, pp. 239–250.

[58] SOSÍK P., *Parallel Accepting Colonies and Neural Networks*, in Păun G. & Salomaa A. (eds.) *Grammatical Models of Multi-Agent Systems*, Gordon and Breach, London, 1999.

[59] SOSÍK P., ŠTÝBNAR L., *Grammatical Inference of Colonies*, in Păun G. & Salomaa A. (eds.) *New Trends in Formal Languages*, Springer, Berlin, 1997.

[60] SRINIVAS B., *Explanation-based Learning and Finite State Transducers: Applications to Parsing Lexicalized Tree Adjoining Grammars*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[61] VILAR J.M., VIDAL E., AMENGUAL J.C., *Learning Extended Finite State Models for Language Translation*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[62] WATSON B.W., *Implementing and using finite automata toolkits*, in Kornai A. (ed.), *Extended Finite State Models of Language*, *Proceedings of the ECAI 96 Workshop*, Cambridge University Press, 1999.

[63] WINTNER S., *Finite-State Technology as a Programming Environment*, in Gelbukh A. (ed.), *CICLing 2007*, LNCS 4394, Springer, Berlin, 2007, pp. 97–106.

[64] YLI-JYRA A., KARTTUNEN L., KARHUMAKI J. (eds.), *Finite State Methods and Natural Language Processing*, Springer, Berlin, 2006.