proceedings

Proceeding Paper

Information Processing by Selective Machines [†]

Mark Burgin * and Karthik Rajagopalan

Department of Computer Science, University of California, Los Angeles (UCLA), Los Angeles, CA 90095, USA; karthikrajagopalan.223@gmail.com

* Correspondence: markburg@cs.ucla.edu

† Presented at the 13th International Workshop on Natural Computing (IWNC), IS4SI Summit 2021, online, 12–19 September 2021.

Abstract: The goal of this paper is to develop the novel automaton model of learning processes called a selective machine and to study the properties of these machines. The model is based on the analysis of the process of language acquisition by people, although it correctly reflects how learning occurs in nature when animals, birds and even fish learn. A selective machine is an abstract automaton that has processors that can belong to different classes of conventional abstract automata. This creates various classes of selective machines. It is proved that in a general case, a selective machine can have higher learning abilities than any of its processors. This shows how synergy emerges in the technological sphere and explains why computer networks are able to outperform separate computers.

Keywords: natural computing; information processing; correction; selection; processor; language; synergy

check for updates

Citation: Burgin, M.; Rajagopalan, K. Information Processing by Selective Machines. *Proceedings* **2022**, *81*, 122. https://doi.org/10.3390/proceedings2022081122

Academic Editors: Marcin J. Schroeder, Masami Hagiya, Yasuhiro Suzuki and Gordana Dodig-Crnkovic

Published: 19 April 2022

1. Introduction

Learning is an important category of information acquisition. Machine learning utilizes automata for learning in general and language learning, in particular. In addition, abstract automata are used for modeling learning by people. In this work, analyzing how people learn natural languages, we develop a new approach to modeling and performing language learning by abstract automata. This allows treating natural language learning as natural computing.

The conventional models of natural language acquisition assume that in the process of learning, children, as well as adult learners, find out and memorize the correct words, rules of generating sentences, and rules of their utilization. However, this picture misses an important peculiarity of the learning process. Namely, people also gain knowledge of incorrect words and sentences, and this knowledge helps them avoid incorrect linguistic constructions in communication.

To model this process, we introduce a new type of computational automata called selective machines. A selective machine can not only generate (compute) words and texts but also eliminate (uncompute) words and texts. This property allows achieving higher power and lower complexity in computations.

It is necessary to remark that this approach to learning has been previously studied in the context of formal grammars [1–6]. Here, we explore learning as a natural information acquisition process, modeling it with computing automata.

2. Constructing Selective Machines

A *selective machine* M has positive and negative processors which accept/recognize words.

The difference between positive and negative processors is their purpose of computation. Positive processors accept or recognize tentative (or possible) elements of a language. However, it is not assumed that all of them are correct (belong to the language under construction). The goal of negative processors is to recognize those elements that do not belong to the language under construction, that is, are incorrect. This allows building a language by the procedure where at first tentative (or possible) elements of the language are extracted, and then the incorrect words are eliminated.

We note that a language L is accepted or recognized by a conventional automaton (machine) M, such as a finite automaton or a Turing machine, if this automaton accepts all words from L and only these words. It is denoted by L_M or $L(M)$ and is also called that language of the machine M.

In the case of selective machines, we have two types of languages.

The *positive language* $L(M_P)$ of the selective machines M is the language accepted/recognized by all positive processors of M.

The negative language is defined in a similar way.

The *negative language* $L(M_N)$ of the selective machines M is the language rejected/eliminated/prohibited by any of the negative processors of M.

Positive and negative languages together recognize the language of selective machines in the following way.

The language $L(M) = L(M_P)\backslash L(M_N)$ is the language of the selective machine M.

3. Classes of Selective Machines

Taking two classes **K** and **H** of automata (algorithms), we denote by **K/H** the class of all selective machines, in which the positive processors are automata from **K** and the negative processors are automata from **H**.

In what follows, we consider selective machines that have one positive processor and one negative processor, each of which belongs to one of the following classes:

- **FA** is the class all finite automata working with words in a given alphabet
- **PA** is the class all pushdown automata working with words in a given alphabet
- **TM** is the class all Turing machines working with words in a given alphabet

This gives us the following classes of selective machines:

1. **FA/FA** is the class of all *finite selective machines*, or FF-selective machines, in which both positive and negative processors are finite automata.
2. **FA/PA** is the class of all FP-selective machines, in which the positive processor is a finite automaton, and the negative processor is a pushdown automaton.
3. **FA/TM** is the class of all FT-selective machines, in which the positive processor is a finite automaton, and the negative processor is a Turing machine.
4. **PA/PA** is the class of all pushdown selective machines, in which both the positive and negative processors are pushdown automata.
5. **PA/FA** is the class of all PF-selective machines, in which the positive processor is a pushdown automaton, and the negative processor is a finite automaton.
6. **PA/TM** is the class of all PT-selective machines, in which the positive processor is a pushdown automaton, and the negative processor is a Turing machine.
7. **TM/TM** is the class of all *Turing selective machines*, or TM-selective machines, in which both positive and negative processors are Turing machines.
8. **TM/FA** is the class of all PF-selective machines, in which the positive processor is a Turing machine, and the negative processor is a finite automaton.
9. **TM/PA** is the class of all TP-selective machines, in which the positive processor is a Turing machine, and the negative processor is a pushdown automaton.

4. Properties of Selective Machines

We note that the recognizing linguistic power RL(A) (RL(**Q**)) of an automaton A (a class **Q** of automata) is the class of all formal languages recognized by the automaton A (by the automata from the class **Q**).

Taking two classes **K** and **H** of automata (algorithms), we can compare their recognizing power. The recognizing power of **K** is larger than or equal to the recognizing power of **H** if $L(\mathbf{H}) \subseteq L(\mathbf{K})$. It is denoted by $\mathbf{H} \leq_L \mathbf{K}$. The recognizing power of **K** is larger than the recognizing power of **H** if $L(\mathbf{H}) \subset L(\mathbf{K})$. It is denoted by $\mathbf{H} <_L \mathbf{K}$.

Lemma 1. *(a) If $\mathbf{D} \leq_L \mathbf{K}$, then $\mathbf{D/H} \leq_L \mathbf{K/H}$.*
(b) If $\mathbf{D} \leq_L \mathbf{H}$, then $\mathbf{K/D} \leq_L \mathbf{K/H}$.

Theorem 1. *The selective (recognizing) power of the class **FA/FA** of finite selective machines is the same as the selective (recognizing) power of the class **FA** of all finite automata.*

Proof. To achieve better understanding of selective machines, we give two proofs of this theorem—one direct and another by reduction to formal grammars. □

(1) *Direct proof*. Let us consider the empty class $\wedge_{\mathrm{FA}}$ of finite automata, i.e., By Lemma 1, $\mathbf{FA} \leq_L \mathbf{FA/FA}$ because $\mathbf{FA} = \mathbf{FA}/\wedge_{\mathrm{FA}}$ and we need only to prove $\mathbf{FA/FA} \leq_L \mathbf{FA}$. Finite automata recognize regular languages. The difference of two regular languages is a regular language, which is accepted by a finite automaton. Consequently, any language from the class $L_A(\mathbf{FA/FA})$ belongs to the class $L_A(\mathbf{FA})$, and thus, $\mathbf{FA/FA} \leq_L \mathbf{FA}$, which implies the equality $\mathbf{FA/FA} =_L \mathbf{FA}$.
(2) *Indirect proof*. Finite automata are linguistically equivalent to regular grammars, i.e., $\mathbf{FA} =_L \boldsymbol{G}_3$ where $\boldsymbol{G}_3$ is the class of all regular grammars. Thus, the class **FA/FA** of finite selective machines is linguistically equivalent to the class $\boldsymbol{G}_{33}$ of grammars with prohibition [1]. Then by Theorem 2 from [2], we have $\mathbf{FA/FA} =_L \mathbf{FA}$.

Theorem is proved.

Theorem 2. *The selective (recognizing) power of the class **PA/FA** of selective machines is the same as the selective (recognizing) power of the class **PA** of all pushdown automata.*

Proof is similar to the proof of Theorem 1.

Theorem 3. *The selective (recognizing) power of the class **TM/FA** of selective machines is the same as the selective (recognizing) power of the class **TM** of all Turing machines.*

Proof is similar to the proof of Theorem 1.

Theorem 4. *The selective (recognizing) power of the class **TM/TM** of Turing selective machines is higher than the selective (recognizing) power of the class **TM** of all Turing machines.*

Proof. Turing machines are linguistically equivalent to unrestricted grammars from the Chomsky hierarchy, i.e., $\mathbf{TM} =_L \boldsymbol{G}_0$ where $\boldsymbol{G}_0$ is the class of all unrestricted grammars. Thus, the class **TM/TM** of finite selective machines is linguistically equivalent to the class $\boldsymbol{G}_{00}$ of grammars with prohibition [1]. Then by Theorem 7 from [2], we have $\mathbf{TM/TM} >_L \mathbf{TM}$. □

Theorem is proved.

The statement of Theorem 4 means that selective machines with Turing machines as their processors can do more than Turing machines. Consequently, **TM/TM** is the class of super-recursive algorithms [7]. This, in turn, refutes the Church-Turing Thesis.

Theorems 3 and 4 display relations of selective machines to the arithmetical hierarchy [8,9].

The language (set) L is *Turing selective recognizable* if it is recognized by some machine from the class **TM/TM**.

Theorem 5. *The class **STR** of all Turing selective recognizable languages (sets) contains the union $\Sigma_1 \cup \Pi_1$.*

This result brings us to the problem whether $\mathbf{STR} = \Sigma_1 \cup \Pi_1$. One more interesting problem is to study selective machines with oracles.

Theorem 6. *The accepting (recognizing) complexity in the class* ***TM/TM*** *of selective machines can be much smaller than the accepting (recognizing) complexity in the class* ***TM*** *of all Turing machines.*

This shows that selective machines can be not only more powerful than Turing machines but also more efficient.

5. Conclusions

We described a novel model of abstract automata called a selective machine, demonstrating that in some cases selective machines have the same recognizing (learning) power as their constituents (processors), while in other important cases, such as selective machines with Turing machines as their constituents (processors), they have higher recognizing (learning) power than their constituents (processors).

In addition, selective machines formalize techniques used in proving some important results for Turing machines. An example of such a result is the Friedberg–Muchnik theorem [8,9].

It is possible to ask whether the same automaton can generate words and exclude those that do not belong to the language under construction. The answer is yes, it is possible, but the results proved by the authors demonstrate that in many important cases, two automata—one positive and another negative, which belong to the same class K (for example, both are Turing machines)—can generate, describe, and recognize many more languages than one automaton from this class K (one Turing machine) can. This clearly shows how synergy emerges from the interaction of constituents in a system.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Burgin, M. Grammars with Prohibition and Human-Computer Interaction. In Proceedings of the Business and Industry Simulation Symposium; Society for Modeling and Simulation International: San Diego, CA, USA, 2005; pp. 143–147.
2. Burgin, M. Basic Classes of Grammars with Prohibition, Preprint in Computer Science. *arXiv* **2013**, arXiv:1302.5181.
3. Burgin, M. Grammars with Exclusion. *J. Comput. Technol. Appl.* **2015**, *6*, 56–66.
4. Carlucci, L.; Case, J.; Jain, S. Learning correction grammars. *J. Symb. Logic* **2009**, *74*, 489–516. [CrossRef]
5. Case, J.; Jain, S. Rice and Rice-Shapiro theorems for transfinite correction grammars. *Math. Logic Q.* **2011**, *57*, 504–516. [CrossRef]
6. Case, J.; Royer, J. Program Size Complexity of Correction Grammars in the Ershov Hierarchy. In *12th Conference of Computability in Europe (CiE 2016), Proceedings, Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 7921, pp. 240–250.
7. Burgin, M. *Super-Recursive Algorithms*; Springer: New York, NY, USA; Berlin/Heidelberg, Germany, 2005.
8. Shen, A.; Vereshchagin, N.K. *Computable Functions*; AMS: Providence, RI, USA, 2003.
9. Rogers, H. *Theory of Recursive Functions and Effective Computability*; MIT Press: Cambridge, MA, USA, 1987.