Red-Green P Automata

Bogdan Aman[1], Erzsébet Csuhaj-Varjú[2], Rudolf Freund[3]

[1] Institute of Computer Science, Romanian Academy
Iaşi, Romania, Email: bogdan.aman@gmail.com
[2] Faculty of Informatics, Etvs Loránd University
Budapest, Hungary, Email: csuhaj@inf.elte.hu
[3] Faculty of Informatics, Vienna University of Technology
Vienna, Austria, Email: rudi@emcc.at

Summary. In this short note we extend the notion of red-green Turing machines to specific variants of P automata. Acceptance and recognizability of finite strings by red-green automata are defined via infinite runs of the automaton on the input string and the way how to distinguish between red and green states.

1 Introduction

In this short note we introduce the notion of red-green automata in the area of P systems. Acceptance and recognizability of finite strings by a red-green Turing machine are defined via infinite runs of the automaton on the input string and the way how to distinguish between red and green states; via infinite runs which are allowed to change between red and green states more than once, more than the recursively enumerable sets of strings can be obtained, i.e.,in that way we can "go beyond Turing". Various possibilities how to "go beyond Turing" to be already found in the literature are discussed in [9]; most of the definitions and results for red-green Turing machines are taken from this paper. In the area of P systems, first attempts to do that can be found in [5] and [8]. Computations with infinite words by P automata have been investigated in [4].

Here we focus on the idea of being able to switch between red and green states in P automata, where states are specific properties of a configuration, for example, the occurrence or the non-occurrence of a specific symbol. As for Turing machines, with one change from red to green states, we can accept all recursively enumerable languages. A similar result can easily be obtained for many variants of P automata, especially for the basic model using antiport rules assigned to the skin membrane.

In this note we only focus on the concept of red-green automata for P automata, without giving formal definitions or proofs, as we assume the reader to know the underlying notions and concepts from formal language theory (e.g., see) as well as from the area of P systems (e.g., see). A lot of research topics wait for being

investigated for P automata "going beyond Turing", but as well for the idea of having red and green configurations together with models of P automata which are not computationally complete, as for example dP automata.

2 Red–Green Turing Machines

A Turing machine M is called a *red–green Turing machine* if its set of internal states Q is partitioned into two subsets, Q_r and Q_g, and M operates without halting. Q_r is called the set of red states, Q_g the set of green states.

Red–green Turing machines can be seen as a type of ω-Turing machines on finite inputs with a recognition criterion based on some property of the set(s) of states visited (in)finitely often, in the tradition of ω-automata (see [4]), i.e., we call an infinite run of the Turing machine on input w *recognizing* if and only if

- no red state is visited infinitely often and
- some green states (one or more) are visited infinitely often.

Comment. In the following, "mind change" means changing the color, i.e., changing from red to green or vice versa.

To get the reader familiar with the basic idea of red-green automata, we give a short sketch of the proofs for some well-known results (see [9]):

Theorem 1. *A set of strings L is recognized by a red–green TM with one mind change if and only if $L \in \Sigma_1$, i.e., if L is recursively enumerable.*

Proof. Let L be the set of strings recognized by a red–green TM M with one mind change. Then design a TM that enumerates all possible inputs, simulates and dovetails the computations of M on these inputs, and outputs string w whenever M makes its first mind change (if any) during the computation on w.

Conversely, if $L \in \Sigma_1$ and M is the TM that enumerates L, then design a red–green TM that on input w simulates the computation of M in red but switches to green when w appears in the enumeration. This machine precisely recognizes L. □

2.1 Red-Green Turing Machines – Going Beyond RE

If more mind changes are allowed, the full power of red–green Turing machines is revealed. For example, the complement of a recursively enumerable set L need not be recursively enumerable, too, but it is always red–green recognizable:

Let M' be the TM recognizing L. Then construct a red–green TM M that operates on inputs w as follows: starting in red, the machine immediately switches to green and starts simulating M' on w. If M' halts (thus recognizing w), the machine switches to red and stays in red from then onward. It follows that M precisely recognizes, in fact accepts, the set L. (Acceptance means that for every

word not recognized by the TM it will never make an infinite number of mind changes, i.e., it finally will end up in red.)

The following result characterizes the computational power of red–green Turing machines (see [9]):

Theorem 2. *(i) Red–green Turing machines recognize exactly the Σ_2 sets of the Arithmetical Hierarchy.*

(ii) Red–green Turing machines accept exactly the Δ_2 sets of the Arithmetical Hierarchy.

3 The basic Model of P Automata

The basic model of P automata as introduced in [2] and in a similar way in [3] is based on antiport rules, i.e., on rules of the form u/v, i.e., the multiset u goes out through the membrane and v comes in instead. As it is already folklore, only one membrane is needed for obtaining computational completeness with only one membrane; the input string is defined as the sequence of terminal symbols taken in during a halting computation. Restricting ourselves to P automata with only one membrane as the basic model, we define a P automaton as follows:

A *P automaton* is a construct

$$\Pi = (O, T, w, R)$$

where

- O is the alphabet of objects,
- T is the terminal alphabet,
- w is the multiset of objects present in the skin membrane at the beginning of a computation, and
- R is a finite set of antiport rules.

The strings accepted by Π consist of the sequences of terminal symbols taken in during a halting computation.

Let us cite from [8]:

> "... a super-Turing potential is naturally and inherently present in evolution of living organisms."

In that sense, we now seek for this potential in P automata.

4 Red-Green P Automata

The main challenge is how to define "red" and "green" states in P automata. In fact, states sometimes are considered to simply be the configurations a P automaton may reach during a computation, or some specific elements occurring in a configuration define its state.

Another variant is to consider the multiset applicable to a configuration as its state, which especially makes sense in the case of deterministic systems. Yet then these multisets have to be divided into "red" and "green" ones.

The easiest way to do this is to specify a subset of the rules as green rules, and all multisets consisting of such green rules only constitute the set of all "green" multisets, whereas all the other ones are "red" multisets.

A stronger condition is to divide the set of rules into "red" and "green" and to define the set of "red" and "green" multisets as those which only consist of "red" and "green" rules, respectively. But then the problem arises how to deal with the multisets of rules consisting of rules of both colors.

5 First Results

As is well known, even with the basic model of P automata as defined above we obtain computational completeness by easy simulations of register machines (which themselves are known, even with only two registers, to be able to simulate the actions of a Turing machine). Hence, the following results are direct consequences of the results known for Turing machines:

Theorem 3. *A set of strings L is recognized by a red–green P automaton with one mind change if and only if $L \in \Sigma_1$, i.e., if L is recursively enumerable.*

Theorem 4. *(i) Red–green P automata recognize exactly the Σ_2 sets of the Arithmetical Hierarchy.*

(ii) Red–green P automata accept exactly the Δ_2 sets of the Arithmetical Hierarchy.

Proof. (Sketch) Let TM be a Turing machine and RM be a register machine simulating TM having its set of internal states Q partitioned into two subsets, Q_r and Q_g; TM operates without halting; Q_r is the set of red states, Q_g the set of green states. The register machine can alos colour its states in red and green, but when simulating the actions of TM eventually needs a green and red variant of its states and actions in order to totally stay within the same color as TM when simulating the actions of one computation step of TM. The P automaton $\Pi = (O, T, w, R)$ can simulate the actions of RM very easily, e.g., see Chapter V in [6], without introducing trap symbols, and even in a deterministic way provided

RM is deterministic. The rules in R are of the form qu/pv where q, p are states of RM and u, v are multisets not containing a state symbol. Hence, a configuration can be defined to exactly have the color of the state symbol from RM currently occurring in the skin region. □

One of the main reasons that the proof of the preceding theorems is that easy is based on the fact that the simulation does not need the trick to trap non-wanted evolutions of the system, which is a trick used very often in the area of P systems. Yet this exactly would contradict the basic feature of the red–green automata way of acceptance by looking at *infinite* computations. Fortunately, the basic model of P automata comes along with this nice feature of not needing trap rules for being able to simulate register machines. Only few models of P automata have this nice feature; another variant are P automata with anti-matter, just recentlyintroduced and investigated, see [1].

6 Future Research

Besides investigating the variants of defining "red"/"green", there are various other models of P automata which deserve to be taken into consideration, e.g., dP automata and anti-matter automata.

There are already a lot of strategies and models to be found in the literature how to "go beyond Turing"; some of them should also be of interest to be considered in the P systems area. Thus, a wide range of possible variants to be investigated remains for future research.

References

1. A. Alhazov, B. Aman, R. Freund, Gh. Păun: Matter and anti-matter in membrane systems. *Brainstorming Week in Membrane Computing*, Sevilla, February 2014.
2. E. Csuhaj-Varjú, Gy. Vaszil: P automata or purely communicating accepting P systems, in: *Membrane Computing, International Workshop, WMC-CdeA 2002*, Curtea de Argeş, Romania, August 19-23, 2002, Revised Papers (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, Eds.), *Lecture Notes in Computer Science* **2597**, Springer, 2003, 219–233.
3. R. Freund, M. Oswald: A short note on analysing P systems. *Bulletin of the EATCS* **78**, 2002, 231–236.
4. R. Freund, M. Oswald, L. Staiger: ω-P Automata with Communication Rules. *Workshop on Membrane Computing, 2003*, 203–217, http://dx.doi.org/10.1007/978-3-540-24619-0_15.
5. C.S. Calude, Gh. Păun: Bio-steps beyond Turing. *Biosystems* **77** (2004), 175–194.
6. Gh. Păun, G. Rozenberg, A. Salomaa (Eds.): *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2010.
7. G. Rozenberg, A. Salomaa (Eds.): *Handbook of Formal Languages,* 3 volumes. Springer, 1997.

8. P. Sosík, O. Valík: On Evolutionary Lineages of Membrane Systems, in: R. Freund et al. (Eds.): *WMC 2005, Lecture Notes in Computer Science* **3850** (2006), 67–78.
9. J. van Leeuwen, J. Wiedermann: Computation as an unbounded process. *Theoretical Computer Science* **429** (2012), 202–212.