String-object transduction with Dogmatic P Systems*

José M. Sempere

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
jsempere@dsic.upv.es

Abstract. In this work we approach the translations of strings to strings in the framework of P systems. We use a variant of P systems based on the Central Dogma of Molecular Biology which establishes the transformation of DNA strands into protein products by applying different string transformation such as transductions and transcriptions.

Keywords: string-objects, membrane systems, transducers

1 Introduction

P systems [3] were introduced as a computational framework inspired by the information and biochemical product processing of living cells through the use of membrane communication. In most of the works about P systems, information is represented as multisets of symbols/objects which can interact and evolve according to predefined rules. Nevertheless, the use of strings to represent the information and the use of rules to transform strings instead of multisets of objects have always been present in the literature of this scientific area. So, in his mostly referred book [3], Gh. Păun overviews the use of string rules in P systems.

Recently [4], it has been proposed a variant of P systems with string-objects and a new kind of rules based on the central dogma of molecular biology which sets the framework to obtain protein products from DNA strands by applying, among others, transduction and transcription operations. It has been proved that the new model, so called *Dogmatic* P Systems, is able of simulating iterated translations, so it is able of generating any recursively enumerable language [4].

2 Basic concepts

We will introduce basic concepts on transducers according to [1].

A *transducer* is defined by the tuple $T = (Q, \Sigma, \Gamma, q_0, E, F)$ where Q is a finite set of *states*, Σ is an input alphabet, Γ is an output alphabet, $q_0 \in Q$ is an

* Work supported by the Spanish Ministerio de Educación y Ciencia under project TIN2007-60769

initial state, $F \subseteq Q$ is the set of *final states* and E is a finite set of transitions satisfying $E \subset Q \times \Sigma^* \times \Gamma^* \times Q$. For any transition in the form (q, x, y, p) with $q, p \in Q$, $x \in \Sigma^*$ and $y \in \Gamma^*$, we will write it as $qx \rightarrow yp$. The transduction rule $qx \rightarrow yp$ means that if the finite control is in state q and it reads the substring x then it changes to state p and writes the substring y. We define a *direct transition step* as follows

$$uqxv \vdash uypv \text{ iff } qx \rightarrow yp \in E$$

The reflexive and transitive closure of $\vdash$ will be denoted by $\vdash^*$. The transduction of an input string x will be defined as

$$T(x) = \{y \in \Gamma^* : q_0 x \vdash^* yp, p \in F\}$$

We can extend the previous definition over languages as

$$T(L) = \bigcup_{x \in L} T(x)$$

Finally, we can take the following normal form in transducers: $F = \{q_f\}$, $E \subset Q - \{q_f\} \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times Q - \{q_0\}$.

3 Dogmatic P systems

In this section, we introduce dogmatic P systems that work with string-objects, in a transduction-like approach, as it was defined in [4]. The *Central Dogma* of Molecular Biology is our source of inspiration for the variant of P system which we will propose in the following. It establishes a metaphor of how DNA strands in the living cell are transformed into protein products by means of information storage and transformation [2].

First, we will introduce the region rules to work with.

A *dogmatic* rule is defined as follows

$$u : v_{pos} \rightarrow w_{ad_1, ad_2, \cdots, ad_k}$$

where u, v are strings, $pos \in \{l, r, *\}$ and for all $i : 1 \leq i \leq k$ $ad_i \in \{here, out, in_j\}$. The meaning is the following: Provided that there exist a string object u in the region (we can omit the presence of u), all the worm objects with substring v at position pos (which means, rightmost one (r), leftmost one (l) or arbitrary position ($*$)) change substring v by w and send a copy of the new string-object at the regions defined by ad_i after eliminating the original string-object from the region.

The addressing label in_j, can be directly applied to contiguous regions at the same level. That is, if there exist regions j and i inside the same region, then a rule at region i can send string-objects to region j directly.

We define a *Dogmatic P system* as the following construct

$$\Pi = (V, \mu, A_1, \cdots, A_m, (R_1, \rho_1), \cdots, (R_m, \rho_m), i_0),$$

where all the elements are defined in the usual way with the exception of A_i, $1 \leq i \leq m$ which is a finite set of strings associated with the region i (the *axioms*), R_i, $1 \leq i \leq m$ which is a finite set of *dogmatic rules* over V associated with the ith region, and ρ_i which is a partial order relation over R_i specifying a *priority*.

Initially, the system holds the set of axioms at every region. Then, all the rules are applied over the strings defined at every region. The system halts whenever no rule can be applied at any region.

The language generated by Π is the set of string-objects collected at region i_0. In the case that $i_0 = \infty$, the language is collected in *external mode* as the set of strings in the environment. The language generated by Π is denoted by $L(\Pi)$. Observe that if the language is infinite then the system will never halt so it will add new string-objects to the output region or the environment.

We can enunciate the following result that relates dogmatic P systems and transducers

Theorem 1. Every transducer can be simulated by a dogmatic P system.

Proof. Let $T = (Q, \Sigma, \Gamma, q_0, E, F)$ be a transducer with $Q = \{q_0, \cdots, q_n\}$ and $F = \{q_n\}$, according to the normal form that we have imposed.

Then, we propose the following dogmatic P system

$$\Pi = (V, \mu, A, A_0, \cdots, A_n, (R, \rho), (R_0, \rho_0), \cdots, (R_n, \rho_n), \infty)$$

where

- $V = \Sigma \cup \Gamma \cup \hat{\Gamma} \cup \{\#\}$, where $\hat{\Gamma} = \{\hat{a} : a \in \Gamma\}$.
- $\mu = [[_0]_0, \cdots, [_n]_n]$ (we have omitted a label for the skin region).
- $A_0 = \{\#w\}$, $A = \emptyset$, and for all $i : 1 \leq i \leq n$ $A_i = \emptyset$.
- **type (a) rules:** For every rule $q_0 x \rightarrow y q_j \in P$, with $x \neq \lambda$ and $y \neq \lambda$, we will add the rule $\#x_l \rightarrow \#\hat{y}_{in_j}$ if $q_j \neq q_0$ or the rule $\#x_l \rightarrow \#\hat{y}_{here}$ if $q_j = q_0$ to R_0
- **type (b) rules:** For every rule $q_i x \rightarrow y q_j \in P$, with $x \neq \lambda$ and $y \neq \lambda$, and for every symbol $\hat{b} \in \hat{\Gamma} \cup \{\#\}$ we will add the rule $\hat{b}x_l \rightarrow \hat{b}\hat{y}_{in_j}$ if $q_i \neq q_j$ or the rule $\hat{b}x_l \rightarrow \hat{b}\hat{y}_{here}$ if $q_i = q_j$ to R_i
- **type (c) rules:** For every symbol $a \in \Sigma$ add the rule $a_r \rightarrow a_{here}$ to the region R_n
- **type (d) rules:** For every symbol $\hat{a} \in \hat{\Gamma}$ add the rule $\hat{a}_r \rightarrow a_{out}$ to the region R_n
- **type (e) rules:** Add to R the rules $\{\hat{a}_l \rightarrow a_{here}\}$
- **type (f) rule:** $\#_l \rightarrow \lambda_{out}$

We will explain the rules in the system as follows: type (a) rules start the transduction of the input string w (here, $\#w$) from the initial state. Hence, we use the # symbol as a left delimiter of the string to be transduced. The alphabet $\hat{\Gamma}$ is used to mark the symbols that have been transduced during the transduction process. Type (b) rules simulate the transitions in the transducer.

Observe that we use the address in_j to change the state in the finite control and the address *here* to simulate the transducer loops. Type (c) rules are used to block those strings that arrive to a final state without finishing the translation process. Finally, type (d) rules output the transduced strings that arrive to a final state with a concluded translation process.

The priorities of the rules in regions R_i keep the following order: type (a) rules > type (b) rules > type (c) rules > type (d) rules.

The rules of the skin region are explained as follows: type (e) rules are used to restore the string symbols of the transduced string in order to output the transduced strings according to the alphabet Γ. The rule of type (f) is used to send the transduced string outside the system so we obtain it from the environment. Here, the priorities are: type (e) rules > type (f) rule.

If we input the string $\#w$, then $T(w) \in L(\Pi)$. We can observe that the transitions from T are simulated by the P system by means of the rules of type (a) and (b). If a complete transduction arrives to a final state, then rules of type (d) are applied and the string is sent to the skin region. Finally, the rules of type (e) are applied and the transduced string is sent out without the left mark # by applying the rule of type (f).

Observe that in the proposed model we have avoided the case $qx \to p$ (i.e. $y = \lambda$). In this case, type (a) rules should be written as $\#x_l \to \#_{in_j}$ or $\#x_l \to \#_{here}$. The same holds for type (b) rules where they should be in the form $\hat{b}x_l \to \hat{b}_{in_j}$ or $\hat{b}x_l \to \hat{b}_{here}$.

In the case $q \to yp$ (i.e. $x = \lambda$), rules of type (b) should take the form $\hat{b}a_l \to \hat{b}\hat{y}a_{in_j}$ or $\hat{b}a_l \to \hat{b}\hat{y}a_{here}$ for every symbol $a \in \Sigma$ and $b \in \Gamma$. In addition the rules $\#a_l \to \#\hat{y}a_{in_j}$ or $\#a_l \to \#\hat{y}a_{here}$ should be added in type (b) (this is the case when the transducer has erased some prefixes by using the rules $qx \to p$ which we will discuss next).

We should consider the case when the transducer rule $q \to yp$ is applied at the end of the string. Here, we should add the rules $\hat{a}b_l \to \hat{a}b_{here}$ for every symbols $a \in \Gamma$ and $b \in \Sigma$ (these rules block the strings to avoid that a suffix is added before finishing the complete translation of the string). In addition, with a lower priority, we have the rules $\hat{a}_r \to \hat{a}\hat{y}_{in_j}$ or $\hat{a}_r \to \hat{a}\hat{y}_{here}$ for every symbol $a \in \Sigma$. These rules have the same priority as type (d) rules in region n, so the system non-deterministically outputs the transduced string or keeps on adding new suffixes according to the transducer rule.

We have omitted the case $q \to p$ (i.e. $x = y = \lambda$) given that we can obtain transducers with no such kind of rules in a normal form.

References

1. Berstel, J: Transductions and Context-Free Languages. Teubner-Verlag (1979)
2. Cohen, W.W.: A Computer Scientist's Guide to Cell Biology. Springer (2007)
3. Păun, Gh.: Membrane Computing. An Introduction. Springer (2002)
4. Sempere, J.M.: *Dogmatic* P Systems. BWMC 2010 (*in press.*)