(P Systems with) Prescribed Teams of Rules Working on Different Objects

Artiom Alhazov Rudi Freund
[Sergiu Ivanov] [Sergey Verlan]

Institute of Math. and Computer Science
Chișinău, Moldova

TU Wien, Austria

Paris-Saclay University, Évry, France
LACL, Univ. Paris Est Creteil, France

BWMC 2025

Overview

Systems with Prescribed Teams of Rules – Previous Research

Systems with Prescribed Teams of Rules Working on Different Objects

Turing Machines

Main Computational Completeness Result

Related Results

Prescribed Teams of Rules on Linear Membrane Structures

Future Research

Systems with Prescribed Teams of Rules

- Artiom Alhazov, Rudolf Freund, Sergiu Ivanov, Sergey Verlan: Tissue P Systems with Vesicles of Multisets. *Int. J. Found. Comput. Sci.* **33** (3&4, pp. 179–202, 2022. DOI = 10.1142/S0129054122410015

- Artiom Alhazov, Rudolf Freund, Sergiu Ivanov, Sergey Verlan: Prescribed Teams of Rules Working on Several Objects. In: Jérôme Durand-Lose and György Vaszil: Machines, Computations, and Universality – 9th International Conference, MCU 2022, Debrecen, Hungary, August 31 – September 2, 2022, Proceedings. Lecture Notes in Computer Science **13419**, Springer, pp. 27–41, 2022. DOI = 10.1007/978-3-031-13502-6_6.

System with Prescribed Teams of Rules

Definition

A *(homogenous) system with prescribed teams of rules of size s and degree d* is a construct $G = (O, O_T, P, R, A)$ where

- O is a set of objects;
- $O_T \subseteq O$ is a set of *terminal objects*;
- P is a finite set of *rules*, i.e., $P = \{p_i \mid 1 \leq i \leq m\}$, $m \geq 0$, $p_i \subseteq O \times O$;
- $R = \{T_1, \ldots, T_n\}$ is a finite set of sets of rules from P called *prescribed teams*, i.e., $T_i \subseteq P$, $1 \leq i \leq n$, with $|T_i| = s$;
- A is a finite set of d initial objects in O.

Computations in a System with Prescribed Teams

A rule $p \in P$ is called *applicable* to an object $x \in O$ if and only if there exists at least one object $y \in O$ such that $(x, y) \in p$.

A computation in G starts with the initial object being the set A. A computation step means the application of a prescribed team of rules to the current configuration being a set of d objects.

At the beginning of a computation step we first have to choose a suitable team T_k.

Computations in a System with Prescribed Teams Working in the Parallel Derivation Mode

The team T_k can only be applied if:

- every rule in T_k can be applied;
- each rule in T_k must be applied to a *different* object in the current set of objects.

The size of a team cannot exceed the degree of the system.

Computations in a System with Prescribed Teams Working in the Sequential Derivation Mode

The team T_k can only be applied if:

- every rule in T_k can be applied one after the other;
- each rule in T_k may be applied to any object in the set of objects obtained after each sequential step.

Several rules of a team can sequentially be applied to the same string.

Derivation Steps in a System with Prescribed Teams

A *derivation step* in G using T_k in the derivation mode δ can be written as $\{O_1, \ldots, O_d\} \Longrightarrow_{T_k,\delta} \{O'_1, \ldots, O'_d\}$ where $\{O_1, \ldots, O_d\}$ is the current set of objects and $\{O'_1, \ldots, O'_d\}$ is the set of objects obtained after the application of T_k in the derivation mode δ.

The derivation relation $\Longrightarrow_{G,\delta}$ of the system G in the derivation mode δ then is the union of all derivation relations $\Longrightarrow_{T_k,\delta}$, $1 \leq k \leq n$.

The reflexive and transitive closure of $\Longrightarrow_{G,\delta}$ is denoted by $\Longrightarrow^*_{G,\delta}$.

Turing Machine

Definition

A *Turing machine* is a construct $M = (Q, V, T_1, T_2, \delta, q_0, q_1, Z_0, B)$ where

- Q is a finite set of *states*,
- V is the *tape alphabet*,
- $T_1 \subseteq V \setminus (\{Z_0, B\})$ is the *input alphabet*,
- $T_2 \subseteq V \setminus (\{Z_0, B\})$ is the *output alphabet*,
- $\delta \subseteq (Q \times V) \to (Q \times V \times \{L, R\})$ is the *transition function*,
- q_0 is the *initial state*, q_1 is the *final state*,
- $Z_0 \in V$ is the *left boundary marker*,
- $B \in V$ is the *blank symbol*.

Turing Machines

A *configuration* of M can be written as Z_0uqvB^ω, where $u, v \in (V \setminus (\{Z_0\}))^*$ and B^ω indicates the remaining infinite empty part of the tape, offering an unbounded number of tape cells initially carrying the blank symbol B.
The current state $q \in Q$ is written to the right of the tape cell on which the read-write head of the Turing machine currently stands.

Main Result

Theorem

The computations of a Turing machine M can be simulated by a homogenous string system with prescribed teams of size 2 and degree 2 using only rules of the form $aI_R(b)$ and $D_R(b)$, either working in the parallel or the sequential derivation mode.

This result is optimal with respect to the size and the degree of the systems, because with string systems of degree 1 with prescribed teams of arbitrary size or systems of arbitrary degree and prescribed teams of size 1 working in the *sequential* or the *parallel* derivation mode, we can only get at most regular languages.

Proof Idea of the Main Theorem

Let $M = (Q, V, T_1, T_2, \delta, q_0, q_1, Z_0, B)$ be a Turing machine. Considering configurations of a Turing machine M as finite strings, we use a right end marker Z_1 to mark the end of a finite representation $Z_0uqvB^mZ_1$ of the configuration Z_0uqvB^ω, where m may be any natural number ≥ 0; m depends on how far on the tape the read-write head has already proceeded during a computation.

$Z_0uqvB^mZ_1$ can be represented by two strings

Z_0uq and $Z_1B^mv^R$.

For simulating the computations of M we construct a system G with prescribed teams of size 2 working on two strings, which represent the part of the Turing tape to the left and the right of the state, also encoding the current state q at the end of (one of) these two strings.

Proof Idea of the Main Theorem

The second string is represented in the primed alphabet; in a team $\{r_1, r_2\}$ then one rule works in the given alphabet, the other one in the primed alphabet; hence, the result is the same both in the sequential and the parallel derivation mode.

Then for any halting computation of M

$$Z_0 w_{input} q_0 B^\omega \Longrightarrow_M^* Z_0 w_{output} q_1 B^\omega$$

we can obtain a corresponding halting computation in G

$$\{Z_0 w_{input} q_0, {Z_1}'\} \Longrightarrow_G^* \{Z_0 w_{output}, {Z_1}' {B'}^k {q_1}'\}.$$

A result $Z_0 w_{output}$ obtained in G represents the string w_{output}.

Observe that Z_0 cannot be avoided as only non-empty strings can be handled by the insertion rules in G.

Systems With Left Insertions and Deletions

Theorem

The computations of a Turing machine M can be simulated by a homogenous string system with prescribed teams of size 2 and degree 2 using only rules of the form $aI_L(b)$ and $D_L(b)$, either working in the parallel or the sequential derivation mode.

Again this result is optimal with respect to the size and the degree of the systems, because with string systems of degree 1 with prescribed teams of arbitrary size or systems of arbitrary degree and prescribed teams of size 1 working in the *sequential* or the *parallel* derivation mode, we can only get at most regular languages.

Systems With Left AND Right Insertions and Deletions

Theorem

The computations of a Turing machine M can be simulated by a homogenous string system with prescribed teams of size 2 and degree 1 using only rules of the form $aI_\alpha(b)$ and $D_\alpha(b)$, $\alpha \in \{L, R\}$, working in the sequential derivation mode.

Again this result is optimal with respect to the size of the systems; moreover, this result obviously can only hold for the sequential derivation mode.

Systems of Size 1 or Degree 1

In the parallel derivation mode, degree 1 also implies size 1, hence, we only get sequential systems.

In the sequential derivation mode, we obtain a characterization of the family of regular languages with systems of degree 1 and size k, $k \geq 3$, whereas otherwise, with degree 1 and size $k < 3$ or with systems of size 1 and arbitrary degree, we cannot even generate all regular languages.

We observe that in case of (sequential) systems of size 1, the resulting language is just the union of the languages obtained by the results of degree 1 with each of the axioms and the teams of rules of size 1.

Linear Membrane Structures

We now consider linear membrane structures as objects and specific membrane rules on these objects.

Definition

Given an alphabet V, a linear membrane structure over V is a sequence $[\cdots[\,[\,]_{a_1}\,]_{a_2}\cdots]_{a_n}$, with $a_i \in V$, $1 \leq i \leq n$, $n \geq 1$.

This linear membrane structure can simply be interpreted as the string $a_1 a_2 \cdots a_n$. The set of all linear membrane structures over V is denoted by $LM(V)$.

Prescribed Teams of Outer Rules in Membrane Systems

Definition

The outer insertion-deletion rules on linear membrane structures are of the form

- $aI_O(b)$ applied to the linear membrane structure $[\cdots[\,]_{a_1}\cdots]_{a_n}$ yields $[[\cdots[\,]_{a_1}\cdots]_{a_n}]_b$, which corresponds to $aI_R(b)$ on the string $a_1\cdots a_n$ yielding $a_1\cdots a_n b$;
- $D_O(b)$ applied to the linear membrane structure $[[\cdots[\,]_{a_1}\cdots]_{a_n}]_b$ yields $[\cdots[\,]_{a_1}\cdots]_{a_n}$, which corresponds to $D_R(b)$ on the string $a_1\cdots a_n b$ yielding $a_1\cdots a_n$.

Prescribed Teams of Inner Rules in Membrane Systems

Definition

The inner insertion-deletion rules on linear membrane structures are of the form

- $aI_I(b)$ applied to the linear membrane structure $[\cdots[\,]_{a_1}\cdots]_{a_n}$ yields $[\cdots[[\,]_b]_{a_1}\cdots]_{a_n}$, which corresponds to $a_1 I_L(b)$ on the string $a_1\cdots a_n$ yielding $ba_1\cdots a_n$;
- $D_I(b)$ applied to the linear membrane structure $[\cdots[[\,]_b]_{a_1}\cdots]_{a_n}$ yields $[\cdots[\,]_{a_1}\cdots]_{a_n}$, which corresponds to $D_L(b)$ on the string $ba_1\cdots a_n$ yielding $a_1\cdots a_n$.

Prescribed Teams of Rules in Membrane Systems

Definition

A *P system with prescribed teams of simple insertion-deletion rules on linear membrane structures* is a construct $\Pi = (LM(V), LM_T(V), P, R, A, d)$ where:

- $LM(V)$: set of linear membrane structures;
- $LM_T(V) \subseteq LM(V)$: set of *terminal linear membrane structures*;
- P: finite set of *outer and/or inner insertion-deletion rules*;
- $R = \{T_1, \cdots, T_n\}$: finite set of *prescribed teams* over P;
- A: finite set of initial objects in $LM(V)$;
- $d \in \{parallel, sequential\}$ is the derivation mode.

The computations in P system with prescribed teams of simple insertion-deletion rules on linear membrane structures are defined as in the general model.

Prescribed Teams of Rules in Membrane Systems

The proofs for the main results nearly verbatim can follow the proofs of the corresponding theorems for strings:

- strings are replaced by linear membrane structures;
- the string insertion rule $aI_R(b)/aI_L(b)$ is replaced by the rule on linear membrane structures $aI_O(b)/aI_I(b)$;
- the string deletion rule $D_R(b)/D_L(b)$ is replaced by the rule on linear membrane structures $D_O(b)/D_I(b)$.

Theorem

The computations of a Turing machine M can be simulated by a homogenous P system with prescribed teams of simple insertion-deletion rules on linear membrane structures of size 2 and degree 2 using only outer or only inner insertion and deletion rules.

Prescribed Teams of Rules in Membrane Systems

Corollary

The computations of a Turing machine M can be simulated by a homogenous P system with prescribed teams of simple insertion-deletion rules on linear membrane structures of size 2 and degree 1 using both outer and inner insertion and deletion rules.

In all cases, we may also include all initial linear membrane objects in a surrounding skin membrane and apply the membrane insertion and deletion rules as rules assigned to the skin membrane, thereby also changing the label of the skin membrane.

Future Research For Systems With Prescribed Teams Working on Several Objects of Various Types

- Systems with prescribed teams working on several objects can also be defined for other objects than strings, for example, multisets, vesicles of multisets, d-dimensional arrays, graphs.
- In all cases, only insertion and deletion rules with contexts of suitable form can be used.
- Moreover, we may also consider other forms of rules, e.g., context-free/non-cooperative rules.

Future Research For Systems With Prescribed Teams Working on Several Objects of Various Types

- (Tissue) P systems with prescribed teams working on several objects can also be defined for other objects than strings or linear membrane structures, for example, multisets, vesicles of multisets, d-dimensional arrays, graphs.
- Various derivation modes well-known in the area of P systems with various forms of rules area to be considered.
- We may also consider polarizations as in our previous paper.

THANK YOU VERY MUCH!

MUCHAS GRACIAS

a nuestros colegas de Sevilla por organizar las sesiones de “brainstorming”.