Testing Identifiable Kernel P Systems using an X-machine Approach

Marian Gheorghe[1], Florentin Ipate[2], Raluca Lefticaru[1,2], Ana Turlea[2]

[1]University of Bradford, [2]University of Bucharest

19th Int. Conference on Membrane Computing
4-7 September, 2018 - Dresden, Germany

Motivation - P Systems Testing

Why Testing?

- Testing is finding out how well something works.
- Under certain conditions, one needs to ensure that the implementation of a system conforms to its specification.

Why P system testing?

- Membrane computing: very fast growing field
- Rapid development of many tools and P system simulators
- This issue raises the problem of testing all these implementations of P systems.
- The models are complex: non-deterministic, parallel, can have polarizations (charges), transformation - communication rules, membrane creation/division etc.

Motivation - P Systems Testing

Previous approaches

- Finite state based testing of P systems
- Mutation-based testing of P systems (Kripe structure)
- Testing non-deterministic stream X-machine models and P system
- Assessing the fault-detection efficiency of previous techniques – mutation testing!
- Automating P system test generation
- Test Generation for P Systems using model checking

Testing Identifiable kP systems

- We present a testing approach for *kP systems* that ensure that the implementations of Identifiable kP Systems conform with their specifications.
  - based on the X-machine method
  - having as core concept the *identifiability* of multisets of rules.

Outcomes of the testing method

For any finite set of computation steps one can get the following:

- for any initial multiset & any two configurations one can get the unique sequence of multisets of rules used to get from one configuration to the other one (the sequence is given by the state cover and uniqueness by identifiability) & each multiset in the sequence is minimal
- any configuration is uniquely identified by a multiset of rules applied to it

Finite Cover Automata

Definition

A finite automaton is a tuple $A = (V, Q, q_0, F, h)$, where:

- V is the finite input alphabet;
- Q is the finite set of states;
- $q_0 \in Q$ is the initial state;
- $F \subseteq Q$ is the set of final states;
- $h : Q \times V \to Q$ is the next-state function.

Definition

Let $A = (V, Q, q_0, F, h)$ be a FA, $U \subseteq V^*$ a finite language and l the length of the longest sequence(s) in U. Then A is called a *deterministic finite cover automaton* (*DFCA*) of U if $L_A \cap V[l] = U$. A *minimal* DFCA - having the least number of states.

Test Suite Definition

Let A be a minimal DFCA for L_A.
The testing method needs two sets of input sequences, S and W:

Definition

$S \subseteq V^*$ is called a proper state cover of A if for every state q of A there exists $s \in S$ such that $h(q_0, s) = q$ and $|s| = level(q)$.

Definition

$W \subseteq V^*$ is called a strong characterisation set of A if for every two states q_1 and q_2 of A and every $j \geq 0$, if q_1 and q_2 are $V[i]$-distinguishable then q_1 and q_2 are $(W \cap V[i])$-distinguishable.

Test Suite

$Y_k = SV[k+1](W \cup \{\lambda\}) \cap V[l] \setminus \{\lambda\}$

X machines Testing

Definition (X machines)

An *X-machine* (XM) is a tuple $Z = (Q, X, \Phi, H, q_0, x_0)$ where:

- Q is a finite set of states;
- X is the (possible infinite) data set;
- Φ is a finite set of distinct *processing functions*; a processing function is a non-empty (partial) function of type $X \to X$;
- H is the (partial) *next-state* function, $H : Q \times \Phi \to Q$;
- $q_0 \in Q$ is the initial state;
- $x_0 \in X$ is the initial data value.

X machines Testing

Definition

Φ is called *identifiable* if for all $\phi_1, \phi_2 \in \Phi$, whenever there exists $x \in X$ such that $\phi_1(x) = \phi_2(x)$, $\phi_1 = \phi_2$.

Definition

A sequence $\phi_1, \ldots, \phi_n \in \Phi^*$, with $\phi_i \in \Phi, 1 \leq i \leq n$, is said to be controllable if there exist $x_1, \ldots x_n \in X$ such that $\phi_i(x_{i-1}) = x_i, 1 \leq i \leq n$. A set $P \subseteq \Phi^*$ is called controllable if for every $p \in P$, p is controllable.

X machines Testing

Let us assume we have an X-machine specification Z and an implementation that behaves like an element Z' of a fault model. In this case, the fault model will be a set of X-machines with the same data set X, type Φ and initial data value x_0 as the specification.

Idea - reduce checking that the IUT Z' conforms to the specification Z to checking that the associated automaton of the IUT conforms to the associated automaton of the X-machine specification.

X machines Testing

Let t be a test transformation of Z and $p = \phi_1 \dots \phi_n$, with $\phi_1, \dots, \phi_n \in \Phi$.

- Suppose p is controllable and let $x_1, \dots, x_n \in X$ such that $\phi_i(x_{i-1}) = x_i, 1 \leq i \leq n$.
  - If $p \in L_{A_Z}$, then $t(p) = x_0 \dots x_n$.
  - If $p \notin L_{A_Z}$, then $t(p) = x_0 \dots x_{k+1}$, where $0 \leq k \leq n-1$, is such that $\phi_1 \dots \phi_k \in L_{A_Z}$ and $\phi_1 \dots \phi_k \phi_{k+1} \notin L_{A_Z}$.
- If p is not controllable, then t(p) is not defined.

A conformance test suite for Z

$$T_k = t(S\Phi[k+1](W \cup \{\lambda\}) \cap \Phi[l] \setminus \{\lambda\}),$$

where S is a proper state cover of A_Z, W is a strong characterisation set of A_Z and t is a test transformation of Z.

Kernel P Systems

Definition (Kernel P systems)

A *kP system* of degree n is a tuple $k\Pi = (A, \mu, C_1, \ldots, C_n, i_0)$, where

- A is a finite set of elements called objects;
- μ defines the membrane structure, which is a graph, (V, E), where V is a set of vertices representing components (compartments), and E is a set of edges, i. e., links between components;
- $C_i = (t_i, w_{i,0})$, $1 \leq i \leq n$, is a compartment of the system consisting of a *compartment type*, t_i, from a set T and an initial multiset, $w_{i,0}$ over A; the type $t_i = (R_i, \rho_i)$ consists of a set of evolution rules, R_i, and an execution strategy, ρ_i;
- i_0 is the output compartment where the result is obtained.

Identifiable transitions in kP systems

Definition

Two rules $r_1 : x_1 \to y_1\{g_1\}$ and $r_2 : x_2 \to y_2\{g_2\}$ from R_1, are said to be *identifiable* if there is a configuration c where they are applicable and if $c \Longrightarrow^{r_1} c'$ and $c \Longrightarrow^{r_2} c'$ then $b(r_1) = b(r_2)$.

According to the above definition the rules r_1 and r_2 are identifiable in c if when the result of applying them to c is the same then their bodies, $x_1 \to y_1$ and $x_2 \to y_2$, are identical.

Identifiable transitions in kP systems

Definition

The multisets of rules $M', M'' \in R_1^*$, are said to be *identifiable*, if there is a configuration c where M' and M'' are applicable and if $c \Longrightarrow^{M'} c'$ and $c \Longrightarrow^{M''} c'$ then $b(M') = b(M'')$.

A kP system $k\Pi$ has its *rules identifiable* if any two multisets of rules, $M', M'' \in R_1^*$, are identifiable.

Example

Considering the rules $r_1 : a \to x\{\geq a\}$, $r_2 : b \to y\{\geq b\}$, $r_3 : a \to y\{\geq a\}$, $r_4 : b \to x\{\geq b\}$, and the configuration ab it is clear that the multisets of rules $M' = r_1 r_2$ and $M'' = r_3 r_4$ are not identifiable in the configuration $c = ab$, as $c = ab \Longrightarrow^{M'} c' = xy$ and $c = ab \Longrightarrow^{M''} c' = xy$, but $b(M') \neq b(M'')$.

Kernel P Systems Testing

- An *automata* model needs to be constructed first, based on the computation tree of the kP system.
- The computation tree may be *infinite* so an *approximation of the tree* is used.
- This approximation is obtained by limiting the length of any computation to an upper bound k and considering only computations up to k transitions in length.
- This approximation is then used to construct a deterministic finite cover automaton (DFCA) of the model

Testing approach using an one-membrane kP system

- the model - $k\Pi = (V, T, \mu_1, w_1, R_1, 1)$.
- X-machine - $Z^t = (Q^t, X, \Phi, H^t, q_0^t, x_0)$ - corresponding to the computation tree of $k\Pi$.
- considering only computations of maximum l steps, where $l > 0$ is a predefined integer.
- $R_1 = \{r_1, \ldots, r_n\}$ - the set of rules of $k\Pi$.
- As only finite computations are considered, for every rule $r_i \in R_1$ there will be some N_i such that, in any step, r_i can be applied at most N_i times, $1 \leq i \leq n$.

X-machine $Z^t = (Q^t, X, \Phi, H^t, q_0^t, x_0)$:

- Q^t is the set of nodes of the computation tree of maximum l steps;
- q_0^t is the root node;
- X is the set of multisets with elements in V;
- x_0 is the initial multiset w_1;
- Φ is the set of (partial) functions induced by the application of multisets of rules $r_1^{i_1} \dots r_n^{i_n}$, $0 \leq i_1 \leq N_1, \dots, 0 \leq i_n \leq N_n$, $i_1 + \dots i_n > 0$;
- H^t is the next-state function determined by the computation tree.

The set of (partial) functions, Φ, from the above definition is identifiable if and only if the corresponding multisets of rules are pairwise identifiable.

Example

The model

$k\Pi_1 = (V, V, \mu_1, w_1, R_1, 1)$, one compartment kP system , where $V = \{a, b, c\}$, $w_1 = ab$, and

$$R_1 = \left\{ \begin{array}{ll} r_1 : a \to b\{\geq a \wedge \geq b\} & r_2 : ab \to bc\{\leq a \wedge \geq b\} \\ r_3 : c \to b\{\geq b \wedge \leq c^{100}\} & r_4 : c \to cc\{\leq c^{100}\} \end{array} \right\}$$

Any two of the above multisets of rules are identifiable.

Example cont.

Computation tree (considering that rules are applied in the maximally parallel mode).

- level 0 - $w_1 = ab$
- level 1 - $ab \Longrightarrow^{r_1} b^2$ and $ab \Longrightarrow^{r_2} bc$.
- level 2 - bc: $bc \Longrightarrow^{r_3} b^2$ and $bc \Longrightarrow^{r_4} bc^2$.
- level 3 - bc^2: $bc^2 \Longrightarrow^{r_3^2} b^3$, $bc^2 \Longrightarrow^{r_3 r_4} b^2c^2$ and $bc^2 \Longrightarrow^{r_4^2} bc^4$.
- level 4 - $b^2c^2 \Longrightarrow^{r_3^2} b^4$, $b^2c^2 \Longrightarrow^{r_3 r_4} b^3c^2$, $b^2c^2 \Longrightarrow^{r_4^2} b^2c^4$, $bc^4 \Longrightarrow^{r_3^4} b^5$, $bc^4 \Longrightarrow^{r_3^3 r_4} b^4c^2$, $bc^4 \Longrightarrow^{r_3^2 r_4^2} b^3c^4$, $bc^4 \Longrightarrow^{r_3 r_4^3} b^2c^6$ and $bc^4 \Longrightarrow^{r_4^4} bc^8$.

Example cont.

Let the upper bound on the number of computation steps considered be $l = 4$.
For this value of l, the rules r_1 and r_2 have been applied at most once, so $N_1 = 1$ and $N_2 = 1$, whereas rules r_3 and r_4 have been applied at most four times, so $N_3 = 4$ and $N_4 = 4$.
Therefore the type Φ of the X-machine Z^t corresponding to the computation tree is the set of partial functions induced by the multisets $r_1^{i_1} r_2^{i_2} r_3^{i_3} r_4^{i_4}$, $0 \leq i_1 \leq 1, 0 \leq i_2 \leq 2, 0 \leq i_3 \leq 4, 0 \leq i_4 \leq 4$, $i_1 + i_2 + i_3 + i_4 > 0$.

Example cont.

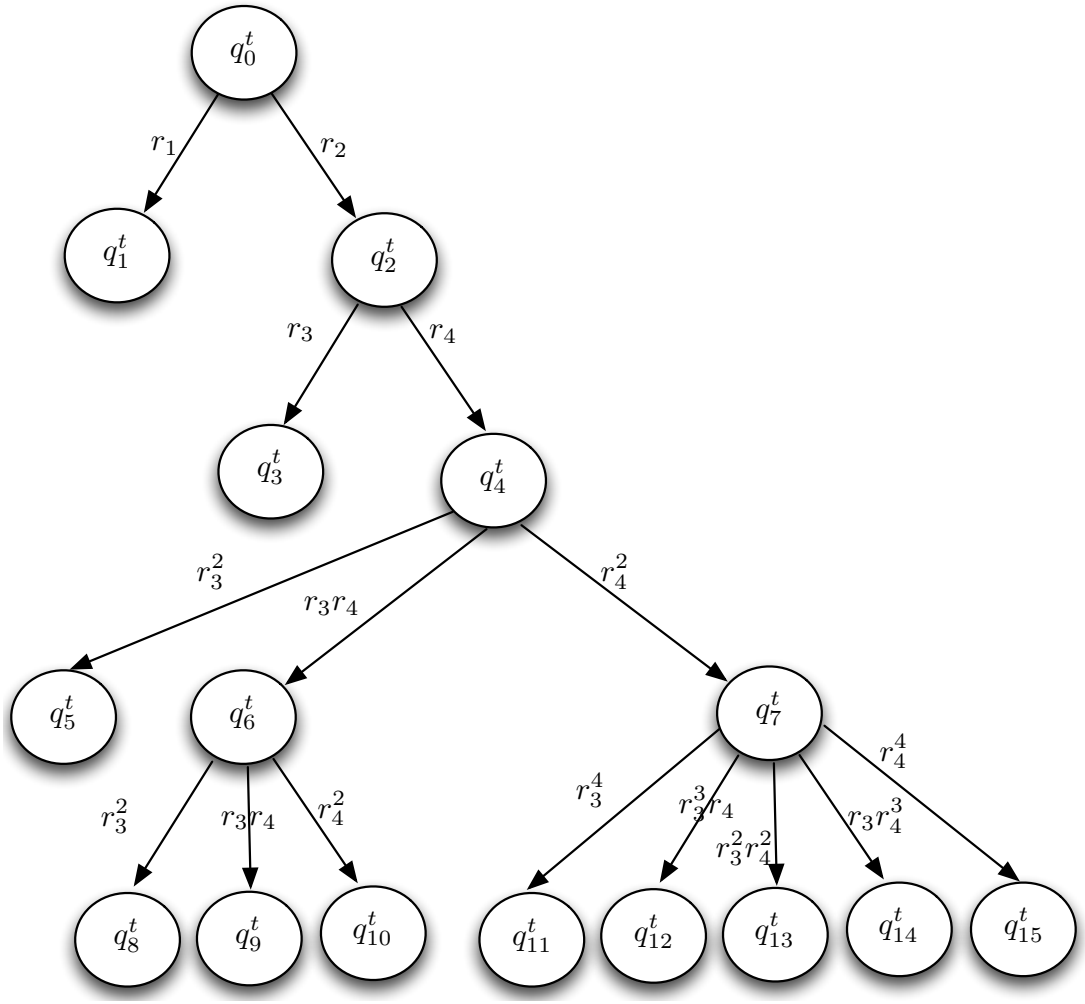

Figure: The associated automaton A_{Z^t} corresponding to the computation tree for $k\Pi_1$ and $l = 4$

DFCA construction

Let $\leq$ be a total order (based on the level from the tree) on Q^t such that $q_1 \leq q_2$ whenever $level(q_1) \leq level(q_2)$ and denote $q_1 < q_2$ if $q_1 \leq q_2$ and $q_1 \neq q_2$.
Define $P^t = \{q \in Q^t \mid \neg\exists q' \in Q^t \cdot q' \sim q, q' < q\}$ and $[q] = \{q' \in Q^t \mid q' \sim q \wedge \neg\exists q'' \in P^t \cdot q'' \sim q', q'' < q\}$ for every $q \in P^t$ (i.e. $[q]$ denotes the set of all states q' for which q is the minimum state similar to q').

Theorem

Let $Z = (Q, X, \Phi, H, q_0, x_0)$, where $Q = \{[q] \mid q \in P^t\}$, $q_0 = [q_0^t]$, $H([q], \phi) = [H^t(q, \phi)]$ for all $q \in P^t$ and $\phi \in \Phi$. Then A_Z is a minimal deterministic finite cover automaton (DFCA) for $L_{A_{Z^t}}$.

Example cont.

Consider Z^t as in the previous example.

$P^t = \{q_0^t, q_1^t, q_2^t, q_4^t, q_7^t\}$; $[q_0^t] = \{q_0^t, q_8^t, q_9^t, q_{10}^t, q_{11}^t, q_{12}^t, q_{13}^t, q_{14}^t, q_{15}^t\}$, $[q_1^t] = \{q_1^t, q_3^t, q_5^t\}$, $[q_2^t] = \{q_2^t\}$, $[q_4^t] = \{q_4^t, q_6^t\}$, $[q_7^t] = \{q_7^t\}$. Then $Z = (Q, X, \Phi, H, q_0, x_0)$, where $Q = \{[q_0^t], [q_1^t], [q_2^t], [q_4^t], [q_7^t]\}$ and $q_0 = [q_0^t]$. The associated automaton of Z is a minimal DFCA for $L_{A_{Z^t}}$ and is as represented in Figure 2.

Example cont.

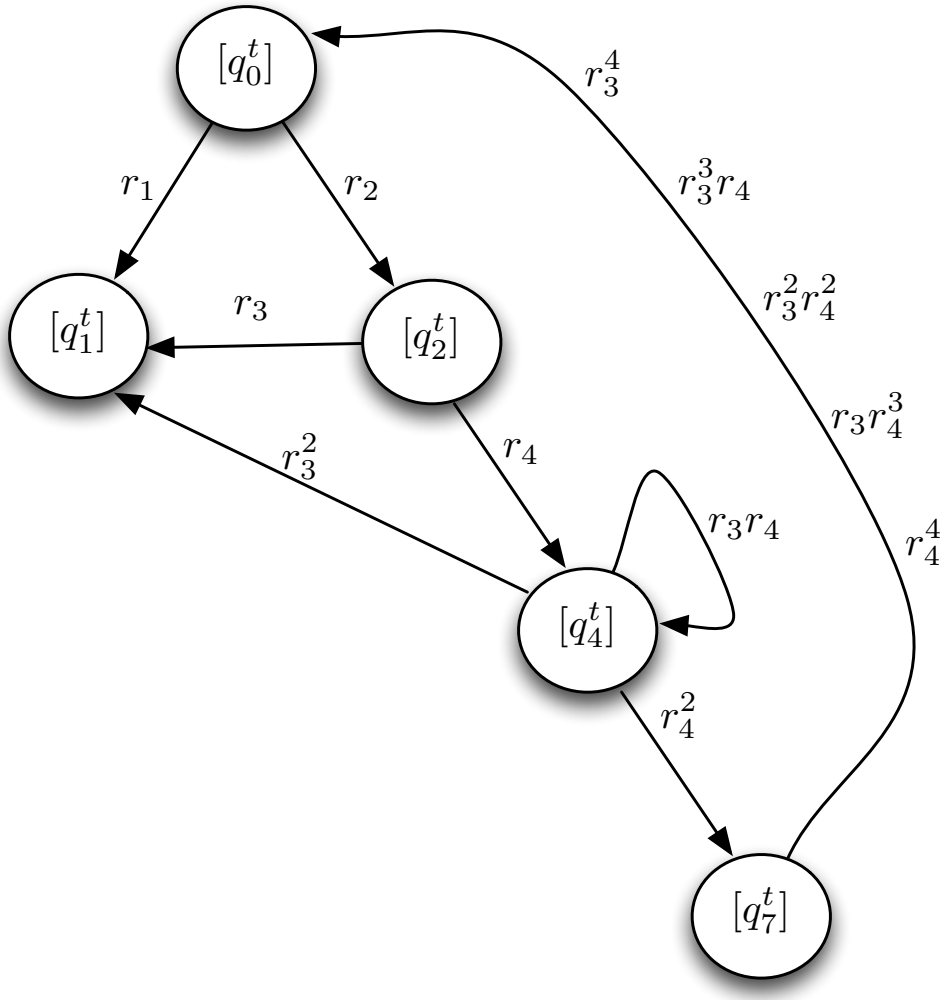

Figure: The DFCA for $L_{A_{Z^t}}$

Example cont.

Test generation process

- λ, r_1, r_2, $r_2\ r_4$, $r_2\ r_4\ r_4^2$ are the sequences of minimum length that reach $[q_0^t], [q_1^t], [q_2^t], [q_4^t]$ and $[q_7^t]$, respectively.
- $S = \{r_1, r_2, r_2\ r_4, r_2\ r_4\ r_4^2\}$ is a proper state cover of Z.
- r_1 distinguishes $[q_0^t]$ from all remaining states and r_3, $r_3 r_4$ and r_3^4 hold the same property for $[q_2^t]$, $[q_4^t]$ and $[q_7^t]$, respectively $\Longrightarrow$ $W = \{r_1, r_3, r_3 r_4, r_3^4\}$ is a strong characterisation set of Z.
- The test suite is given by the formula $T_k = t(Y_k)$, where $Y_k = S\Phi[k+1](W \cup \{\lambda\}) \cap \Phi[l] \setminus \{\lambda\}$ and t is a test transformation of Z.

Conclusions and Future Work

- Testing approach for kernel P systems that, under certain conditions, ensures that the implementation conforms to the specification.
- The methodology is based on the *identifiable kernel P systems* concept, which is essential for testing.
- Has been introduced for one-compartment kP systems with rewriting rules, but could be extended.
- Future work: planning to show how more complex engineering problems can be solved, tested and verified by using kP systems.

Questions and Answers

Thank you!