УДК 004.414.23.042

I.D. Perepelytsya, G.M. Zholtkevych

V.N. Karazin Kharkiv National University, Kharkiv

HIERARCHIC DECOMPOSITION OF PRE-MACHINES AS MODELS OF SOFTWARE SYSTEM COMPONENTS

Abstract. *In this paper we consider a model of system interaction with the environment, which allows us to forget its full protocol from time to time. Mathematical models of such interactions are pre-machines. We showed that the machines can be completely described in terms of some special class of digraphs. An approach to analyzing the structure of pre-machines in terms of their derivatives series is offered.*

Keywords: *finite state pre-machine, prefix marked digraph, region, interval, derivative digraph, derivatives series, hierarchic decomposition*

1. Introduction

The development of global networking technology attracts attention of researchers and software developers to the problem of interaction of remote network nodes.

In this context, the key problem that arises is the problem of minimizing the amount of data about the current session state of information exchange between client and server. Unfortunately, the use of stateless strategy [1], which is the least expensive, is impossible for most applications. Thus, there is the task of organizing information exchange, which allows "to forget" the full information about the session state from time to time. This behaviour on the server side is modelled by pre-automata, which were introduced in [2, 3].

This paper is concerned with defining an approach to the structural analysis of pre-automata, which has shown its effectiveness in solving problems of control-flow analysis of computer programs [4].

These results were presented at the first international seminar "Specification and Verification of Hybrid Systems" [5].

2. Preliminaries

Let A and B be any sets, then

(a) the set of all total maps from A to B is denoted by $[A \to B]$;

(b) the set of all partial maps from A to B is denoted by $(A \to B)$;

(c) if $f \in (A \to B)$ and $a \in A$ then the statement "$f(a)$ is not defined" is denoted by $f(a) = \varnothing$, and the statement "$f(a)$ is not defined" is denoted by $f(a) \neq \varnothing$.

Let Σ be a finite alphabet, then

(a) the free semi-group generated by Σ is denoted by Σ^+;

(b) the free monoid generated by Σ is denoted by Σ^*;

(c) a subset P of Σ^* is called a prefix code if for $u, v \in \Sigma^*$ from the conditions $uv \in P$ and $u \in P$ it follows $v = \varepsilon$;

(d) for a word $w \in \Sigma^+$ let us denote by $px(w)$ the subset of Σ^+ such that $u \in px(w)$ if $w = uv$, $u \neq \varepsilon$, and $v \neq \varepsilon$.

A lot of graph theory notions are used in the paper. But it is known, that in the field some authors use the same term with different meanings, and some authors use different terms to mean the same thing (compare [6, 7, 8] for example). Therefore, we need to set some notation before stating our results.

Definition 1. Let $V(\mathbf{G})$ and $E(\mathbf{G})$ are finite sets of vertices and edges respectively, iv and tv are maps from $E(G)$ to $V(\mathbf{G})$ then a **digraph** $\mathbf{G}$ is a tuple $(V(\mathbf{G}), E(\mathbf{G}), iv, tv)$ such that the following statement is true

$$(\forall e_1, e_2 \in E(\mathbf{G}))\ (iv(e_1) = iv(e_2) \wedge tv(e_1) = tv(e_2) \Rightarrow e_1 = e_2)$$

For some digraph $\mathbf{G}$ we shall say that an edge $e \in E(\mathbf{G})$ connects the **initial vertex** $iv(e) \in V(\mathbf{G})$ with the **terminal vertex** $tv(e) \in V(\mathbf{G})$.

If $iv(e) = v_1$ and $tv(e) = v_2$ then we use the following notation: $v_1 \xrightarrow{e} v_2$.

Definition 2. Let $\mathbf{G}$ be a digraph. A **walk** in $\mathbf{G}$ is an alternating sequence of vertices and edges $\alpha = v_0, e_1, v_1, \ldots, e_n, v_n$ beginning and ending with a vertex and such that $v_{i-1} \xrightarrow{e_i} v_i$ for all $i = 1, \ldots, n$.

In this case n is called a **length** of the walk α.

We can consider walks with length 0 and identify them with vertices. We shall use the notation $\varepsilon_v = v$ for such walks.

For a walk $\alpha = v_0, e_1, v_1, e_2, \ldots, e_n, v_n$ we shall use the notation

$$\alpha = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} v_n .$$

Definition 3. Suppose α and β be walks in the digraph $\mathbf{G}$ such that

$$\alpha = v_0^\alpha \xrightarrow{e_1^\alpha} v_1^\alpha \xrightarrow{e_2^\alpha} \ldots \xrightarrow{e_m^\alpha} v_m^\alpha ,$$

$$\beta = v_0^\beta \xrightarrow{e_1^\beta} v_1^\beta \xrightarrow{e_2^\beta} \ldots \xrightarrow{e_n^\beta} v_n^\beta , \text{ and } v_m^\alpha = v_0^\beta ,$$

then the alternating sequence

$$\gamma = v_0^\alpha \xrightarrow{e_1^\alpha} v_1^\alpha \xrightarrow{e_2^\alpha} \ldots \xrightarrow{e_m^\alpha} \boxed{v_m^\alpha = v_0^\beta} \xrightarrow{e_1^\beta} v_1^\beta \xrightarrow{e_2^\beta} \ldots \xrightarrow{e_n^\beta} v_n^\beta$$

is a walk in the digraph $\mathbf{G}$ which is called **concatenation** of the walks α and β. In this case the walk γ is denoted by $\alpha\beta$.

3. Finite State Pre-machines

In this section a generalization of finite-state machines will be introduced. Corresponding class of objects is known [2] as the class of finite pre-automata or the class of finite-state pre-machines (in abbreviated form FSPM).

Definition 4. A **finite state pre-machine** is the triple (Σ, X, T), where Σ is a finite alphabet, X is a finite set of states, $T \in \left(X \times Z^* \to X\right)$, and the following conditions hold:

(a) $\varnothing \neq T(x, \varepsilon) = x$ for any $x \in X$;

(b) for any $u, v \in \Sigma^*$ and $x \in X$ from $T(x,u) \neq \varnothing$ and $T\left(T(x,u), v\right) \neq \varnothing$ it follows that

$$\varnothing \neq T(x, uv) = T\left(T(x,u), v\right);$$

(c) for any $w \in \Sigma^+$ and $u, v \in \Sigma^+$ such that $w = uv$ and $x \in X$ from $T(x,w) \neq \varnothing$ and $T(x,u) \neq \varnothing$ it follows that

$$\varnothing \neq T\left(T(x,u), v\right) = T(x,w).$$

To describe the behaviour of a pre-machine a notion of a pre-machine's snapshot is needed.

Definition 5. Let $\mathbf{M} = (\Sigma, X, T)$ be a finite state pre-machine, then an element $(x,w) \in X \times \Sigma^*$ is called a **snapshot** if the following condition holds $T(x,u) = \varnothing$ where $\varepsilon \neq u \in \mathrm{p}x(w)$.

The set of all snapshots is denoted by $\mathbf{S}(\mathbf{M})$.

For a finite state pre-machine $\mathbf{M} = (\Sigma, X, T)$ let us construct a total map $\overline{T} \in \left[\mathbf{S}(\mathbf{M}) \times \Sigma \to \mathbf{S}(\mathbf{M})\right]$ using the partial map T by the formula

$$\overline{T}\left((x,w), a\right) = \begin{cases} (x, wa), & T(x, wa) = \varnothing \\ \left(T(x, wa), \varepsilon\right), & T(x, wa) \neq \varnothing \end{cases}$$

As it is proved in [2], the triple $\left(\Sigma, \mathbf{S}(\mathbf{M}), \overline{T}\right)$ is an automaton.

We can interpret a finite state pre-machine $\mathbf{M} = (\Sigma, X, T)$ as a control device with a buffer that fires on the buffer contents changing and its response is determined by the current state and the buffer contents:

1) elements of X are interpreted as states of this device;
2) each snapshot (x,w) describes the current state of the device and its buffer contents;
3) elements of he alphabet Σ describe external signals which the device receive;
4) the map $\overline{T}$ describes a device's snapshot changing as a response to a signal.

Note that similar device can be used as a dispatcher of heterogeneous handling of flows of events [3].

The dynamics of the control device is described in Table 1.

Table 1

Interaction between a pre-machine and an environment

An external influence	**A pre-machine response**
	1) Initialize a pre-machine: buffer $\leftarrow \varepsilon$ active_state $\leftarrow$ initial_state
	2) Wait a signal
3) The signal $a \in \Sigma$ has been sent	
	4) Append a into the pre-machine's buffer: buffer $\leftarrow$ (buffer)a
	5) If $T(\text{active\_state}, \text{buffer}) \neq \varnothing$ then active_state $\leftarrow$ $T(\text{active\_state}, \text{buffer})$ buffer $\leftarrow \varepsilon$
	6) go to item 2)

One can consider a map $T \in (X \times \Sigma \to X)$ as the map $\hat{T} \in \left[X \to \left(\Sigma^* \to X\right)\right]$ that is defined by the following formula

$$\hat{T}(x)(w) = T(x,w).$$

Using this notation we redefine the class of pre-machines.

Definition 6. A finite state pre-machine is a triple $\left(\Sigma, X, \hat{T}\right)$, where Σ is a finite alphabet, X is a finite set of states, $\hat{T} \in \left[X \to \left(\Sigma^* \to X\right)\right]$, and for the triple the following conditions hold:

(a) for any $u, v \in \Sigma^*$ and any $x', x'' \in X$ from $\varnothing \neq \hat{T}(x')(u) = x''\$$ and $\hat{T}(x'')(v) \neq \varnothing$ it follows that $\varnothing \neq \hat{T}(x')(uv) = \hat{T}(x'')(v)$;

(b) for any $u, v \in \Sigma^*$ and any $x', x'' \in X$ from $\varnothing \neq \hat{T}(x')(u) = x''$ and $\hat{T}(x')(uv) \neq \varnothing$ it follows that $\varnothing \neq \hat{T}(x'')(v) = \hat{T}(x')(uv)$.

It is evident, that Definitions 4 and Definition 6 are equivalent.

4. Pre-machines and Prefix Marked Digraphs

In this section relationships between the class of finite state pre-machines and some subclass of digraphs are studied.

First let us define the subclass of digraphs mentioned above.

Definition 7. A triple $(\Sigma, \mathbf{G}, P)$ where Σ is a finite alphabet, $\mathbf{G}$ is a digraph, and $P \in \left[E(\mathbf{G}) \to \mathbf{2}^{\Sigma^+}\right]$ is called a **prefix marked digraph** (PMD) if the following conditions hold

(a) $\bigcup_{e \in E(\mathbf{G}): iv(e) = v} P(e)$ is a prefix code for all $v \in V(\mathbf{G})$;

(b) any $e_1, e_2 \in E(\mathbf{G})$ such that $iv(e_1) = iv(e_2)$ and $P(e_1) \cap P(e_2) \neq \varnothing$ are equal.

Definition 8. Let $(\Sigma, \mathbf{G}, P)$ be a PMD, $w \in \Sigma^*$, and $\alpha = v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} v_n$ be a walk in $\mathbf{G}$.

The walk α **carries the word** w if there exits decomposition $w = w_1 \ldots w_n$ such that $w_i \in P(e_i)$ for all $i = 1, \ldots, n$.

The next simple proposition is very important.

Proposition 1. Suppose, that walks α and β have the same initial vertex and they carry the same word w then they are equal.

Now we can construct for arbitrary PMD $(\Sigma, \mathbf{G}, P)$ some finite state pre-machine $(\Sigma, \mathbf{G}, P)^\dagger = \left(\Sigma, V(\mathbf{G}), \hat{T}\right)$.

Let us define $\hat{T}$ by the following way:

1) define $\hat{T}(v)(\varepsilon) = v$ for all $v \in V(\mathbf{G})$;
2) $\hat{T}(v)(w) \neq \varnothing$ if there exists the walk α such that it carries the word w and its initial vertex is v (uniqueness of α follows from Proposition 1);
3) in this case terminal vertex v' of the walk α is uniquely determined by the vertex v and the word w, so the following definition is correct:

$$\hat{T}(v)(w) = v'.$$

Proposition 2. The triple $\left(\Sigma, V(\mathbf{G}), \hat{T}\right)$ is a finite state pre-machine.

Let us describe the inverse construction: for an arbitrary finite state pre-machine $\mathbf{M} = \left(\Sigma, X, \hat{T}\right)$ construct some PMD $\left(\Sigma, X, \hat{T}\right)^\dagger = (\Sigma, \mathbf{G}, P)$.

Let us define:

1) $V(\mathbf{G}) = X$;
2) $E(\mathbf{G}) = \left\{(x_1, x_2) \in X^2 \mid \left(\exists w \in \Sigma^+\right)\left(\varnothing \neq \hat{T}(x_1)(w) = x_2\right)\right\}$;
3) for $e = (x', x'') \in E(\mathbf{G})$ determine $iv(e) = x'$ and $tv(e) = x''$;
4) $P(e) = \left\{w \in \Sigma^+ \mid \varnothing \neq \hat{T}(it(e))(w) = tv(e) \wedge \left(\forall u \in px(w)\right)\left(T(it(e))(w) = \varnothing\right)\right\}$.

It is evident that the triple $\left(\Sigma, X, \hat{T}\right)^\dagger = (\Sigma, \mathbf{G}, P)$ is a PMD.

Note that one can easily prove that

firstly, prefix marked graphs $(\Sigma, \mathbf{G}, P)$ and $\left((\Sigma, \mathbf{G}, P)^\dagger\right)^\dagger$ match up to notation;

secondly, finite state pre-machines $\left(\Sigma, X, \hat{T}\right)$ and $\left(\left(\Sigma, X, \hat{T}\right)^\dagger\right)^\dagger$ match up to notation.

This results show that structures of PMDs and FSPMs are interdependent.

5. Unextendable Sets of Vertices in Digraphs

To study the structure of digraphs we consider some class of their vertices subsets.

In this section we suppose that $\mathbf{G} = (V, E, it, vt)$ is some digraph.

Definition 9. A subset A of the set V is an **unextendable set** if the following condition holds

$$\left(\forall v \in tv\left(iv^{-1}(A)\right) \setminus A\right)\left(iv\left(tv^{-1}(v)\right) \not\subset A\right).$$

Proposition 3. Let $\mathbf{U}(\mathbf{G})$ be a family of all unextendable subsets of V, then $\mathbf{U}(\mathbf{G})$ is a Moore family [9, p. 111, Definition].

As known [9, p. 111, Theorem 1], each Moore family is a family of closed subsets for some uniquely defined closure operator.

Let us denote by $\mathrm{Hull}^{\mathbf{G}}[A]$ the closure of a subset A with respect to the closure operator corresponding to $\mathbf{U}(\mathbf{G})$. We shall say that $\mathrm{Hull}^{\mathbf{G}}[A]$ is the **unextendable hull** of the set A.

Definition 10. A map $\mathrm{Inf}^{\mathbf{G}}: 2^V \to 2^V$ is called an **inflation** if it is defined by the formula

$$\mathrm{Inf}^{\mathbf{G}}[A] = A \cup \left\{ v \in \mathrm{tv}\left(\mathrm{iv}^{-1}(A)\right) \setminus A \mid \mathrm{iv}\left(\mathrm{tv}^{-1}(v)\right) \subset A \right\}.$$

We claim that for any digraph its family of unextendable sets can be described as the fixed-points set of the corresponding inflation. To show this we need some properties of inflations.

Proposition 4. Let $\mathrm{Inf}^{\mathbf{G}}: 2^V \to 2^V$ be the inflation then the following properties hold:

(a) $\mathrm{Inf}^{\mathbf{G}}[\varnothing] = \varnothing$;

(b) $A \subset \mathrm{Inf}^{\mathbf{G}}[A]$ for any $A \subset V$;

(c) if $A_1 \subset A_2 \subset V$ then $\mathrm{Inf}^{\mathbf{G}}[A_1] \subset \mathrm{Inf}^{\mathbf{G}}[A_2]$.

Proposition 5. Suppose $\mathrm{Inf}^{\mathbf{G}}: 2^V \to 2^V$ is the inflation, then any set $A \subset V$ of is unextendable if and only if it is a fixed point of the inflation.

Corollary 1. For each $A \subset V$ there exists the least inflation fixed point such that it covers A. This fixed point equals to the unextendable hull of A.

6. Calculation of Inflations and Unextendable Hulls

The aim of this section is to consider methods to calculate the inflation and the unextendable hull for any subset of vertices for an arbitrary digraph.

First let us describe the algorithm (see Algorithm 1) to calculate the inflation of some subset of vertices.

Algorithm 1. Calculating the inflation of a set

Require: a directed graph $\mathbf{G}$, and a subset A of V

Ensure: $\mathrm{Inf}^{\mathbf{G}}[A]$

```
W ← ∅;
for e ∈ E do
    if iv(e) ∈ A and tv(e) ∉ A then
        W ← W ∪ {tv(e)};
    end
end
/*  The following condition has been true:  */
/*  W = iv(tv^{-1}(A)) \ A                  */
for v ∈ W do
    for e ∈ E do
        if tv(e) = v and iv(e) ∉ A then
            W ← W \ {v};
            break
        end
    end
end;
/*  The following condition has been true:  */
/*  W = {v ∈ iv(tv^{-1}(A)) \ A | tv(iv^{-1}(v)) ⊂ A}  */
return A ∪ W
```

It is clear that the algorithm is correct.

The next theorem is needed to calculate the unextandable hull for any set.

Theorem 1. Let $\mathbf{L}$ be a finite lattice, z be an element of $\mathbf{L}$, and $T: \mathbf{L} \to \mathbf{L}$ be a map such that the following conditions hold

(a) $x \le T(x)$ for any $x \in \mathbf{L}$;

(b) $T(x) \le T(y)$ for any $x, y \in \mathbf{L}$ such that $x \le y$,

then the set $F_z = \{x \in \mathbf{L} \mid z \le x \wedge x = T(x)\}$ is not empty and contains its greatest lower bound.

From the proof of Theorem 1 the next algorithm to calculate of the unextendable hull for any subset of vertices an be obtained.

Algorithm 2. Calculating the unextendable hull of a set

Require: a directed graph $\mathbf{G}$, and a subset A of V

Ensure: $\mathrm{Hull}^{\mathbf{G}}[A]$

```
W_0 ← A;
W_1 ← Inf^G[W_0];
while W_0 ≠ W_1 do
    W_0 ← W_1;
    W_1 ← Inf^G[W_0]
end;
return W_0
```

7. Regions and Intervals

In the section some class of vertices set in a digraph is introduced. The definition of the class is a generalization of the class of regions for a flow graph [10]. This class as a class of intervals was introduced in [4, p. 938]. In the paper we shall distinguish the class of regions and the class of maximal regions which we call intervals.

In this section we suppose that $\mathbf{G} = (V, E, \mathrm{iv}, \mathrm{tv})$ is some digraph.

Definition 11. A subset $A \subset V$ is called a **region** in $\mathbf{G}$ if $A = \mathrm{Hull}^{\mathbf{G}}\left[\{v\}\right]$ for some $v \in V$.

In this case v is called a **header** of a region A.

The next proposition establishes the principal property of regions.

Proposition 6. Let A be a region in $\mathbf{G}$ and $e \in E$ such that $\mathrm{tv}(e) \in A$, but $\mathrm{iv}(e) \notin A$, then $\mathrm{tv}(e)$ is the unique header of A.

Definition 12. A region is called an **interval** if it can not be covered by other region.

Corollary 2 (of Proposition 6). Let I_1 and I_2 be different intervals then their intersection is empty.

One can obtain the following result by summarizing the arguments adduced above.

Theorem 2. The set of intervals for any digraph forms the partition of the set of its vertices.

8. Derivative Finite State Pre-Machines and Hierarchic Decomposition

In this section a procedure that provides pre-machine states aggregating is described. We hope the procedure can be used for hierarchical specification of complex systems with behavior similar to the behavior described by Table 1.

Let $\mathbf{M} = \left(\Sigma, X, \hat{T}\right)$ be a FSPM and $\mathbf{M}^{\dagger} = (\Sigma, \mathbf{G}, P)$ be the corresponding PMD.

Consider the partition $X = I_1 \cup \ldots \cup I_n$, where $I_1, \ldots, I_n$ are intervals of the digraph $\mathbf{M}^{\dagger}$. Denote by $h(I)$ a set of headers for a interval $I \in \{I_1, \ldots, I_n\}$.

Denote by $X^{(1)}$ the set $\{I_1, \ldots, I_n\}$.

For $1 \le k, l \le n$ a pair (I_k, I_l) is included into $E^{(1)}$ if there exists some walk

$$\alpha = x_0 \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n$$

such that

1) $x_0 \in h(I_k)$;
2) $x_1, \ldots, x_{n-1} \in I_k$;
3) $x_n \in I_l$.

If we define $\mathrm{iv}((I_k, I_l)) = I_k$ and $\mathrm{tv}((I_k, I_l)) = I_l$ then we obtain the digraph $\mathbf{G}^{(1)} = \left(X^{(1)}, E^{(1)}, \mathrm{iv}, \mathrm{tv}\right)$.

Define a marking function $P^{(1)} : E^{(1)} \to \Sigma^{+}$ by the next way:

let's include the word w into $P^{(1)}((I_k, I_l))$ if there exists a walk $\alpha = x_0 \xrightarrow{e_1} x_1 \xrightarrow{e_2} \ldots \xrightarrow{e_n} x_n$ such that

1) $x_0 \in h(I_k)$;
2) $x_1, \ldots, x_{n-1} \in I_k$;
3) $x_n \in I_l$;
4) α carries w.

Proposition 7. The triple $\left(\Sigma, \mathbf{G}^{(1)}, P^{(1)}\right)$ is a PMD.

Definition 13. The finite state pre-machine $\left(\Sigma, \mathbf{G}^{(1)}, P^{(1)}\right)^{\dagger}$ is called the **derivative finite state pre-machine** for the FSPM $\mathbf{M}$ and it is denoted by $\mathbf{M}^{(1)}$.

Using Definition 13 one can build the **derivative series** for any FSPM $\mathbf{M}$:

$$\mathbf{M} = \mathbf{M}^{(0)}, \mathbf{M}^{(1)}, \ldots, \mathbf{M}^{(n)}, \ldots,$$

where

$$\mathbf{M}^{(n+1)} = \left(\mathbf{M}^{(n)}\right)^{(1)}, \qquad n = 0, 1, \ldots.$$

Taking into account that all sets of states are finite and quantity of elements in the sets decreases when position of series member increases one can conclude that the series terminates.

The derivative series for FSPMs is similar to the derivative control flow graphs [4] and we hope that they can be used for structural analysis of finite states pre-machines.

9. One sample of hierarchic decomposition

In this section the sample of hierarchical decomposition is presented. We demonstrate that the derivatives series of the model pre-machine has only two members and the last member of it has three states.

In the Fig. 1 the model of semantic events recognizer is presented.

In this context a semantic event is some sequence of elementary events (signals). Of course, not all such sequences are valid. So, the task of the specified software component is to check validity of a sequence and to assign a relevant handler. Undoubtedly, we should not forget to handle errors and timeouts. It is suggested that at arbitrary time point the event "reset command

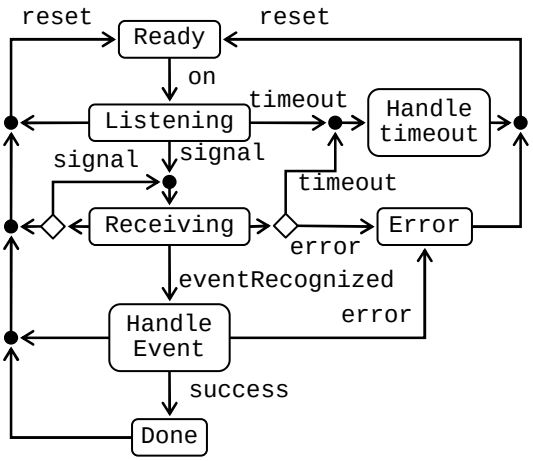

Fig. 1. Semantic events recognizing

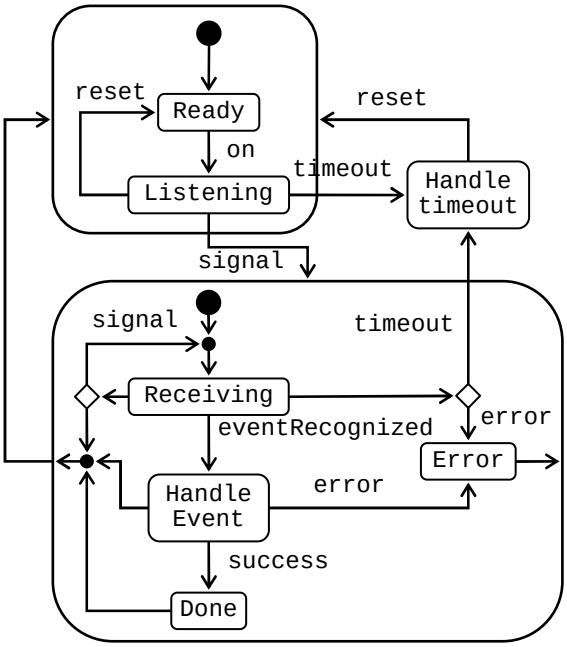

Fig. 2. Derivative pre-machine

has been received" can be fired.

In the Fig. 2 the derivative pre-machine is presented. One can see that its intervals are singletons.

10. Conclusion

Summarizing we can formulate the problems that arise in connection with an introduction to the concepts of the paper:

firstly, to study the case when the last member of a derivative series has only one state.

secondly, to describe intervals of a digraph in the terms of this digraph.

thirdly, to study specific properties of the derivative series for a finite state machine.

fourthly, to propose a method using a pre-machine's derivative series for step-by-step refining of the pre-machine specification.

Bibliography

1. Fowler M. Patterns of Enterprise Application Architecture / M. Fowler. – Boston: Pearson Education, 2003. –

2. Dokuchaev M. Partial actions and automata / M. Dokuchaev, B. Novikov, G. Zholtkevych. – Algebra and Discrete Mathematics. − 2011. – V. 11, No 2. – P. 51 – 63.

3. Novikov B. Pre-automata as Mathematical Models of Event Flows Recognisers / B. Novikov, I. Perepelytsya, G. Zholtkevych // V. Ermolayev et al. (eds.) Proc. 7-th Int. Conf. ICTERI 2011, Kherson, Ukraine, May 4-7, 2011. – CEUR-WS.org/Vol-716, ISSN 1613-0073, 2011. – P. 41 – 50.

4. A.V. Aho. The Theory of Parsing, Translation, and Compiling, Volume 2: Compiling / A.V. Aho, J.D. Ullman // Series in Automatic Computation. – Englewood Cliffs, NJ: Prentice-Hall, Inc, 1973. –

5. Novikov B. Derivatives Series of Finite State Pre-Machines / B. Novikov, I. Perepelytsya, G. Zholtkevych // Specification and Verification of Hybrid Systems. – Proc 1st Int. Seminar, Kyiv, Ukraine, October 10 – 12, 2011. – T. Shevchenko Nat. Univ. in Kyiv, Paul Sabatier Univ. Toulouse, State Found for Fund. Res. Ukraine, 2011. - P. 40 - 50.

6. Bollobas B. Modern Graph Theory / B. Bollobas // Graduate Text in Mathematics. – New York: Springer Science + Business Media, Inc, 1998. –

7. Bondy A. Graph Theory / A. Bondy, U.S.R. Murt // Graduate Text in Mathematics. – Berlin Heidelberg New York: Springer, 2008. –

8. Diestel R. Graph Theory, 3rd ed. / R. Diestel // Graduate Text in Mathematics. – Berlin Heidelberg New York: Springer, 2006. –

9. Birkhoff G. Lattice Theory, 3rd ed. / G. Birkhoff // Colloquium Publications. – Providence, RI: Amer Mathematical Soc, 1995. –

10. Aho, A.V. Compilers: Principles, Technique, and Tools, 2nd ed. / A.V. Aho, M.S. Lam, R. Sethi, J.D. Ullman. – Boston, MA: Pearson Education, Inc, 2007. –

Рецензент: д-р техн.. наук, проф. М.В. Ткачук, Національний технічний університет «Харківський політехнічний інститут», Харків

Автор: ЖОЛТКЕВИЧ Григорій Миколайович
Харківський національний університет імені В.Н. Каразіна, Харків, доктор технічних наук, професор, декан механіко-математичного факультету, завідувач кафедри теоретичної та прикладної інформатики.
Роб. тел. – 707-53-25, E-mail: g.zholtkevych@gmail.com

Автор: ПЕРЕПЕЛИЦЯ Іван Дмитрович
Харківський національний університет імені В.Н. Каразіна, Харків, аспірант кафедри теоретичної та прикладної інформатики.
Роб. тел. – 707-55-35, E-mail: ivanperepelytsya@gmail.com

Ієрархічна декомпозиція перед-машин та її застосування до аналізу програмних систем
Г.М. Жолткевич, І.Д. Перепелиця
У статті розглянуто модель взаємодії програмної системи з навколишнім середовищем, яка дозволяє моделювати забування час від часу повного протоколу. Для цього як математичні моделі використані перед-машини. Показано, що перед-машини можуть бути повністю описані в термінах деякого спеціального класу орграфів. Запропоновано підхід до аналізу структури перед-машин з точки зору їх похідного ряду, що дає метод ідентифікації ієрархічної структури.
Ключові слова: машини зі скінченою множиною станів, префіксно-маркований орграф, регіон, інтервал, похідний орграф, похідний ряд, ієрархічна декомпозиція

Иерархическая декомпозиция пред-машин и ее применение к анализу программных систем
Г.Н. Жолткевич, И.Д. Перепелица
В статье рассматривается модель взаимодействия программной системы с окружающей средой, которая позволяет моделировать забывание время от времени полного протокола. Для этого в качестве математических моделей использованы пред-машины. Показано, что пред-машины могут быть полностью описаны в терминах некоторого специального класса орграфов. Предложен подход к анализу структуры пред-машин с точки зрения их производного ряда, что приводит к методу идентификации иерархической структуры.
Ключевые слова: машины с конечным множеством состояний, префиксно-маркированный орграф, регион, интервал, производный орграф, производный ряд, иерархическая декомпозиция