ON THE CLASS OF L*-LANGUAGE FORMULAS THAT SPECIFY FINITE-MEMORY FINITE-STATE MACHINES

A. N. Chebotarev

The class of $L^*$-language specifications that specify finite-memory finite-state machines is characterized. The transformation of an $L^*$-language specification into an equivalent specification that specifies a finite-memory finite-state machine is substantiated.

Keywords: specification language $L^*$, $\exists$-formula, bi-infinite $\omega$-word, $\Sigma$-automaton, finite-memory finite-state machine, $k$-separability.

INTRODUCTION

Finite-memory finite-state machines (FSMs) [1, 2] play an important role in the theory and practice of designing discrete systems. On the one hand, such FSMs possess properties that allow one to simplify methods of their analysis, transformation, and realization and, on the other hand, practically any realization of an FSM in the form of a hardware module or in the form of a program can be considered as a finite-memory FSM. Moreover, any FSM without finite memory can be transformed into a finite-memory FSM due to an extension of its alphabet with preserving its functionality, i.e., its function. Logical languages of specification of FSMs without finite memory are much richer than languages of specification of finite-memory FSMs. An example of a language allowing for the specification of only finite-memory FSMs is the sufficiently simple language $L$ [3]. The use of this language allows one to considerably simplify design procedures, i.e., the passage from a declarative specification of requirements on the functioning of an FSM to its imperative (procedural) representation. The language $L^*$ is an extension of the language $L$ owing to the introduction of additional descriptive means necessary for the specification of FSMs without finite memory. Note that neither the language $L^*$ nor any other language of first-order predicate logic make it possible to specify arbitrary FSMs. Nevertheless, the language $L^*$ allows one to specify a rather wide class of FSMs that are of interest from the practical viewpoint. An extension of the language $L$ has led to a considerable complication of synthesis methods based on the specification theorem [4]. The version of the language $L^*$ described in [4] was constructed so that formulas of the language could be equivalently transformed to a form satisfying the conditions of the specification theorem. Here, somewhat extended variant of this language is considered that is beyond the scope of the requirements of the theorem and, as a result, available synthesis procedures can produce an incorrect result in some cases. Moreover, the majority of design methods such as checking specifications for consistency, determinizing them, checking an open system for realizability, verification, and others are developed for $L$-language specifications. Therefore, of great importance is the possibility of transformation of specifications from the language $L^*$ into the language $L$. Taking into account that only finite-memory FSMs can be specified in the language $L$, this transformation presumes the passage from the specification of an FSM without finite memory to the specification of the corresponding finite-memory FSM. To substantiate a method of passage to the language $L$, it is necessary to characterize the class of $L^*$-language formulas that specify finite-memory FSMs. It is obvious that all $L$-language formulas belong to this class, but formulas that specify finite-memory FSMs and do not belong to the language $L$ are of special interest. In this article, necessary concepts are introduced and, in
CYCLIC FINITE-MEMORY Σ-AUTOMATA

The languages L and L* are used for the specification and design of reactive systems, in particular, FSMs over infinite words (ω-words). The main object of specification and synthesis is a partial (and in some cases, uninitialized) deterministic FSM without outputs $\mathcal{A} = \langle \Sigma, Q, \delta_A \rangle$, where $\Sigma$ is a finite input alphabet, $Q$ is a finite set of states, and $\delta_A : Q \times \Sigma \to Q$ is a partial transition function. We call such an FSM a $\Sigma$-automaton.

A $\Sigma$-automaton $A = \langle \Sigma, Q, \delta_A \rangle$ is called cyclic if, for each $q \in Q$, there are $\sigma_1, \sigma_2 \in \Sigma$ and $q_1, q_2 \in Q$ such that $q_1 \in \delta_A(q, \sigma_1)$ and $q_2 \in \delta_A(q_2, \sigma_2)$.

The behavior of a cyclic $\Sigma$-automaton can be conveniently described in terms of sets of words and ω-words over the alphabet $\Sigma$, and, therefore, we give main definitions connected with these concepts.

Let $\Sigma$ be a finite alphabet, let $\mathbf{Z}$ be the set of integers, let $N^+ = \{ z \in \mathbf{Z} \mid z > 0 \}$, and let $N^- = \{ z \in \mathbf{Z} \mid z \leq 0 \}$. A mapping $r$ of a set $\{ 1, \ldots, n \}$ into $\Sigma$ is called a word of length $n$ in the alphabet $\Sigma$ and is denoted by $r = r(1)r(2)\ldots r(n)$. Mappings $u : \mathbf{Z} \to \Sigma$, $l : N^+ \to \Sigma$, and $g : N^- \to \Sigma$ are called, respectively, a bi-infinite ω-word (denoted by $\ldots u(2)u(1)u(0)u(-1)u(-2)\ldots$), an ω-word (denoted by $l(1)l(2)\ldots$), and a left-infinite ω-word (denoted by $g(-2)g(-1)g(0)$) in the alphabet $\Sigma$. The set of all words in the alphabet $\Sigma$ is denoted by $\Sigma^*$. We denote the sets of all ω-words and left-infinite ω-words in the alphabet $\Sigma$ by $\Sigma^0$ and $\Sigma^{-0}$, respectively. A segment $u(\tau)(\tau + 1)\ldots(\tau + k)$ of a bi-infinite ω-word $u$ is denoted by $u(\tau, k)$. We call infinite segments $u(-\infty, k)$ and $u(k + 1, \infty)$ a $k$-prefix and a $k$-suffix, respectively, of the bi-infinite ω-word $u$. For $n \in N^+$, an $n$-prefix of an ω-word $l$ is understood to be a word $l(1)\ldots l(n)$ and an $n$-suffix of the left-infinite ω-word $g$ is understood to be a word $g(1)\ldots g(n)$. An ω-word $l = \sigma_1\sigma_2\ldots$ in the alphabet $\Sigma$ is admissible at a state $q$ of the $\Sigma$-automaton $A$ if there is an ω-word of states $q_0q_1q_2\ldots$, where $q_0 = q$, such that, for any $i = 0, 1, 2, \ldots$, we have $q_{i+1} = \delta_A(q_i, \sigma_{i+1})$.

A left-infinite ω-word $\ldots \sigma_{-1}\sigma_0$ in the alphabet $\Sigma$ is representable by a state $q$ of the $\Sigma$-automaton $A$ if there is a left-infinite ω-word of states $\ldots q_2q_1q_0$, where $q_0 = q$, such that, for any $i = -1, -2, \ldots$, we have $q_{i+1} = \delta_A(q_i, \sigma_{i+1})$.

Thus, the following two sets of ω-words are associated with each $q_i$ of the cyclic $\Sigma$-automaton: the set $S(q_i)$ of all ω-words admissible at the state $q_i$ and the set $P(q_i)$ of all left-infinite ω-words representable by the state $q_i$. Similarly, for an arbitrary $k \in N^+$, we consider the set $S^k(q_i)$ of all words of length $k$ admissible at the state $q_i$ and the set $P^k(q_i)$ of all words of length $k$ representable by the state $q_i$.

States $q_i$ and $q_j$ are called equivalent if we have $S(q_i) = S(q_j)$. An FSM without equivalent states is called reduced.

Let the set $Q$ of states of the $\Sigma$-automaton $A$ be $\{ q_1, \ldots, q_n \}$. We call the family of sets of ω-words $(S(q_1), \ldots, S(q_n))$ the behavior of the $\Sigma$-automaton $A$.

We call two FSMs $A_1$ and $A_2$ with behaviors $(S'_1, \ldots, S'_m)$ and $(S''_1, \ldots, S''_m)$, respectively, equivalent if each $S'_i$ $(i = 1, \ldots, n)$ is among $S''_1, \ldots, S''_m$ and each $S''_i$ $(i = 1, \ldots, m)$ is among $S'_1, \ldots, S'_n$.

Let $A = \langle \Sigma, Q, \delta_A \rangle$, let $q_1, q_2 \in Q$, and let $\sigma \in \Sigma$. We call the triple $< q_1, \sigma, q_2 >$ such that we have $\delta_A(q_1, \sigma) = q_2$ a transition in the $\Sigma$-automaton $A$ from the state $q_1$ to the state $q_2$. We say that an input word $r = \sigma_1\ldots\sigma_n$ converts a state $q''$ into a state $q'''$ if, for each $i = 1, \ldots, n$ in the FSM $A$, there is the transition $< q', \sigma_i, q_i+1 >$ and $q' = q_1$ and $q''' = q_{i+1}$.

A $\Sigma$-automaton $A$ is called a finite-memory FSM if there is a natural $k$ such that, for any input word of length $k$, all the states of the FSM $A$ at which it is admissible are converted by this word into equivalent states. Such a minimal $k$ is called the memory depth of the FSM.

**Statement 1.** A reduced cyclic $\Sigma$-automaton $A$ with a set of states $Q$ has a finite memory whose depth is no larger than $k$ if and only if, for any $q_1, q_j \in Q$ ($q_i \neq q_j$), we have $P^k(q_i) \cap P^k(q_j) = \emptyset$.

**Proof. Necessity.** Assume the opposite, i.e., that there are states $q_1, q_j \in Q$ such that we have $P^k(q_i) \cap P^k(q_j) \neq \emptyset$. Then there are states that are converted by the same word of length $k$ into nonequivalent states (by virtue of the reducedness of the automaton), contrary to the definition of a finite-memory FSM of depth $k$. 

2