ON AN AUTOMATON EQUIPPED WITH A MECHANISM OF RECURSIVE CALL

By

Seiiti HUZINO

(Received Oct. 31, 1977)

1°. It is a well known fact that for every context free language there are algorithms which construct a pushdown automaton recognizing the language. One of these is an algorithm by S. Ginsburg, and the other is that by J. E. Hopcroft and J. D. Ullman ([1; p. 63, Lemma 2.5.1], [2; p. 75, Theorem 5.2]). Both algorithms show a way leading to a pushdown automaton having only one state, by making use of forms of right parts of productions of the grammar generating the given language. Of course there is no objection to that fact. However it seems to be a weak point that the automaton thus constructed has a mechanism only depending on operations of pushdown stack and has no mechanism operating on its main body.

In this paper we shall show a new type of automaton with mechanism which reflects enough for structure-characteristics of a context-free language.

2°. Let us show it by an example. Let $X = \{a, b\}$ be an alphabet and consider a language $L = \{a^n b^n | n = 1, 2, \dots\}$ over X . The structure diagram of the language is shown as in figure 1 ([3]).

Omitting arrows leading to the empty set in the diagram, together with the empty set itself, and rewriting symbols ∂_a and ∂_b by a and b , respectively, we get the following infinite automaton which recognizes the language L , by considering the node of each derivative language as state of the automaton (Figure 2).

Here we insert state-numbers 0, 1 and 2 in order to make the following discussion understandable. The state diagram of the automaton consists of two parts, that is, the part of the figure 3 and that of the figure 4, and the original diagram can be obtained by docking the above two parts at the state-nodes 1 and 2.

Furthermore, by vertically rewriting the first segment of the diagram of figure 4 as shown in the dotted segment, we see that it again corresponds to the original

diagram except the assignment to final state. In the other words, it means that the original diagram can be constructed from the diagram in figure 3 by recursively starting from the original one at the node 1 and ending to the original one at the node 2. We shall denote it by the following diagram (Figure 5). Denote the whole diagram by a symbol A , and draw an arrow from the node 1 to the node 2 in the figure 3, and put the symbol A over the arrow.

Fig. 5

Fig. 6

In order to regard it as a state diagram of an automaton, we may consider a new type of mechanism, that is, a mechanism with a pushdown-stack (See figure 6). When the state of the automaton is the state 1 and the input symbol is not b , push down a symbol A into the stack, go back to the initial state and restart. So doing, when the automaton reaches to a final state, pop up the symbol A from the stack, and return back to the state 2, and continue the operations. We consider such mechanism which recursively calls the automaton when the automaton meets a symbol A . This is the first thought which we have obliged to feel the necessity of a mechanism of 'recursive call'.

3°. We shall define an automaton with such mechanism.

DEFINITION 1. A finite automaton with recursive call mechanism over X is a system $\mathfrak{A} = \langle \{\mathfrak{A}_z | z \in Z\}, s_0, F \rangle$ where $\mathfrak{A}$ has a pushdown stack, Z and X are two alphabets disjoint from each other, and $\mathfrak{A}_z$ is formally an incomplete, not-deterministic finite automaton $\langle X \cup Z, S_z, \delta_z, s_0^{(z)}, F_z \rangle$, and the stack operates as follows: When $\delta_z(s, \zeta) = t$ ($\zeta \in Z, s, t \in S_z$) pushdown a pair of symbols $\langle z, \bar{t} \rangle$ (the symbol $\bar{t}$ corresponds to the state t) into the stack, transit the state to the initial state of the automaton $\mathfrak{A}_t$, and continue the operations. And, when

$\delta_z(s, x) \in F_z$ ($x \in X$), look up the topmost part of the stack, and if the content of the stack is empty, terminate the operations of the automaton $\mathfrak{A}_z$, and if the content of the topmost part of the stack is $\langle \tilde{z}, \bar{p} \rangle$ ($\tilde{z} \in Z$, $p \in S_{\tilde{z}}$), pop up it, and continue the operations from the state p of the automaton $\mathfrak{A}_{\tilde{z}}$. Here, when $p \in F_{\tilde{z}}$ and an input sequence has read down, the sequence is accepted by the automaton. The state s_0 , called by the initial state, is an element of the coproduct $\coprod_{z \in Z} S_z$, and $F = \coprod_{z \in Z} F_z$ is called by the set of designated states or final states.

The operations of the automaton $\mathfrak{A}$ start at the initial state of $\mathfrak{A}_{z_0}$, corresponding to the state s_0 , under the situation that the stack of the automaton is empty. By the above mechanism, automata are called from each other, and continue to read the given input sequence w . When the automaton has no function to operate for an input, pushdown a pair of symbols *<the name of the present part of automaton, the symbol of next state used when the operation returns back to the present part of automaton>* into the stack, transit the state to the initial state of an automaton corresponding to possible 'stack' symbol $z \in Z$, and continue the operation. The whole operations are in general not-deterministic. In such a way, if there is a way of some pushdown and pop-up techniques described above, and if the last state transited by the input sequence is a final state in F , it is called that the input sequence w is accepted by the automaton. (See figure 7).

Fig. 7

DEFINITION 2.

$$T_{\text{call}}(\mathfrak{A}) = \{w | w \in X^*, \text{ and } w \text{ is accepted by } \mathfrak{A}\}.$$

4°. We have the following results:

THEOREM 1. *For each ε -free context-free grammar $\mathfrak{G} = \langle Z, X, P, S \rangle$, there exists an automaton $\mathfrak{A}$ with recursive call mechanism such that*

$$T_{\text{call}}(\mathfrak{A}) = \mathfrak{L}(\mathfrak{G}),$$

where $\mathfrak{L}(\mathfrak{G})$ is the language generated by the grammar $\mathfrak{G}$.

PROOF. Let us construct an automaton $\mathfrak{A}_z$ corresponding to a symbol $z \in Z$ as follows: For each production $z \rightarrow \alpha_i$ ($\alpha_i \in (X \cup Z)^+$) put a finite set of states such that

$$\delta_z^*(s_0^{(z)}, \alpha_i) = f^{(z)},$$

where $s_0^{(z)}$ is the initial state of $\mathfrak{A}_z$, $\{f^{(z)}\}$ is a single element set of final state and δ_z^* is an extended dynamics of $\mathfrak{A}_z$, just constructed. The automaton $\mathfrak{A}_z$ thus constructed is an incomplete, not-deterministic finite automaton. Let $\mathfrak{A}$ be an automaton with recursive call mechanism $\langle \{\mathfrak{A}_z | z \in Z\}, s_0, F \rangle$, where s_0 is a state corresponding to the initial state of $\mathfrak{A}_S$ (S being the start symbol of the grammar), and F is the set $\coprod_{z \in Z} F_z$. It is obvious that $T_{\text{call}}(\mathfrak{A}) = \mathfrak{L}(\mathfrak{G})$, by definition of recursive call mechanism of the automaton.

THEOREM 2. *For each automaton $\mathfrak{A}$ with recursive call mechanism there exists a context-free grammar $\mathfrak{G}$ such that $\mathfrak{L}(\mathfrak{G}) = T_{\text{call}}(\mathfrak{A})$.*

PROOF. Let $\mathfrak{A}$ be a system $\langle \{\mathfrak{A}_z | z \in Z\}, s_0, F \rangle$, where

$$\mathfrak{A}_z = \langle X \cup Z, \delta_z, s_0^{(z)}, F_z \rangle,$$

$$F = \coprod_{z \in Z} F_z, s_0 \in \coprod_{z \in Z} S_z,$$

and δ_z is a relation on $S_z \times (X \cup Z) \times S_z$, admitting recursive calling 'by name'. The automaton $\mathfrak{A}_z$ is regarded as an usual finite automaton in the sense of Rabin-Scott over the alphabet $X \cup Z$. Then the language $T(\mathfrak{A}_z)$ recognized by $\mathfrak{A}_z$ is regular over $X \cup Z$. Hence the structure diagram of the language $T(\mathfrak{A}_z)$ is a finite, directed graph. A regular grammar over $X \cup Z$ is automatically constructed so that it generates the regular language recognized by the finite

automaton which is obtained from the above diagram. The grammar itself is regarded as a context-free grammar over X , the symbol z being a start symbol of the grammar. Construct such a grammar for each automaton $\mathfrak{A}_z$. Let the whole set of productions with a start symbol S , which corresponds to the state s_0 , be $\mathfrak{G}$. The grammar $\mathfrak{G}$ is obviously context-free over X . By construction it follows that $T_{\text{call}}(\mathfrak{A}) = \mathfrak{L}(\mathfrak{G})$. Combining with these results, and slightly modifying, we have the following theorem:

THEOREM 3. *A language is context-free if and only if there exists a finite automaton with recursive call mechanism recognizing the language.*

Such a mechanism is useful for considering treatments of languages, in particular, compiling of programming languages.

References

- [1] S. GINSBURG: *The Mathematical Theory of Context-Free Languages*, McGraw-Hill Book Co. (1966), 232 p.
- [2] J. E. HOPCROFT, J. D. ULLMAN: *Formal Languages and their Relation to Automata*, Addison-Wesley Pub. Co. (1969), 242 p.
- [3] S. IMORI, S. HUZINO: *On relationship between a structure diagram of context-free language and a mechanism of pushdown automaton*, Conf. Rec. of Applied Math. of Jap. Math. Soc. (1976, Oct.) 166–171.