Semi-Deterministic Virtual Finite Automaton(SDVFA) of order (s,t) and Regular Grammar

A. Jain

Abstract. In this paper, we establish the equivalence between semi-deterministic virtual finite automaton(SDVFA) of order (s,t) and and regular grammar.

AMS Subject Classification (2000): 68Q45

Keywords: Semi-deterministic virtual finite automaton, formal languages, regular grammar

1. Introduction

In [11], the author introduced the notion of Semi-deterministic virtual finite automaton(SDVFA) of order (s,t) as a generalization of DFA , NFA and VDFA [9]. In this paper, we shall show that every SDVFA of order (s,t) is represented by a regular grammar and for every regular grammar, we can have an SDVFA of appropriate order. In other words, we shall show that given an SDVFA, we can have a Type 3 grammar specifying the language accepted by the SDVFA and given a Type 3 grammar, we can construct an SDVFA that accepts the language specified by the grammar meaning thereby that class of languages accepted by an SDVFA is exactly the class of Type 3 languages or regular languages.

Firstly, we define an SDVFA of order (s,t) [11] and regular grammar [2], [15].

Definition 1.1. A grammar is said to be a **regular grammar** or **Type 3 grammar** ([2], [15]) if the left side of each production is a nonterminal symbol and right side of each production contains atmost one nonterminal symbol, in which case it is the right most symbol of the right side.In the following, we shall use A and B to denote arbitrary nonterminals, a and b to denote arbitrary terminals, and α and β to denote arbitary strings of terminals and nonterminals. Therefore, a grammar is said to be a **Type-3**

grammar or **regular grammar** if all productions in the grammar are of the forms:

$$A \to a$$

or

$$A \to aB. \tag{1.1}$$

In other words, in any production the left-hand string is always a single nonterminal and right-hand string is either a terminal followed by a nonterminal. Furthermore, it can be shown that productions of the form in (1.1) are equivalent to productions of the forms:

$$A \to \gamma$$

or

$$A \to \gamma B$$

where γ is an arbitrary string of terminals including the string of zero length i.e. empty string ϵ.

Definition 1.2. A semi-deterministic virtual finite automaton (SDVFA) of order (s,t) is a finite automaton that can make atmost "s"$(s \geq 1)$ transitions on receiving a real input and atmost "t"$(t \geq 0)$ transitions on virtual input (or no input). (Zero transition means the automaton remains in the same state).

Remark 1.1. For an SDVFA having n states, we have the following:

(i) If $s = 1$ and $t = 0$, then an SDVFA of order $(1,0)$ is simply a DFA [2,6,13,14,15,17,18].

(ii) If $s = 1$ and $t = n$, then an SDVFA of order $(1,n)$ is simply a VDFA [9].

(iii) If $s = n$ and $t = 0$, then an SDVFA of order $(n,0)$ is simply an NFA [2,6,13,14,15,17,18].

(iv) If $s = n$ and $t = n$, then an SDVFA of order (n,n) is simply an ϵ-NFA [2,6,13,14,15,17,18].

We formally define a semi-deterministic virtual finite automaton(SDVFA) of order (s,t) as follows:

Definition 1.3. A semi-deterministic virtual finite automaton (SDVFA) of order (s,t) consists of

1. A finite set of states (including the dead state) often denoted by Q.
2. A finite set of input symbols including the empty string symbol ϵ. This is often denoted by $\Sigma \cup \{\epsilon\}$. Σ is called real alphabet.
3. A transition function $\delta_{(s,t)}$ that takes as arguments a state and an input symbol. On real input symbol i.e. if the symbol is a member of real alphabet Σ, $\delta_{(s,t)}$ returns a set of atmost "s" states while on virtual input ϵ, the transition function returns a set of atmost "t" states.
4. A start state S which is one of the states in Q.
5. A set of final or accepting states F. The set F is a subset of Q. Dead state is never an accepting state and it makes a transition to itself on every possible input symbol.

We can also denote an SDVFA of order (s,t) by a "five tuple" notation:

$$V = (Q, \Sigma \cup \{\epsilon\}, \delta_{(s,t)}, q_0, F)$$

where V is the name of the SDVFA, Q is the set of states, $\Sigma \cup \{\epsilon\}$ is the set of input symbols, $\delta_{(s,t)}$ is the transition function, q_0 is the start state and F is the set of accepting states.

In Section 2 and Section 3, we discuss the construction of regular grammar from an SDVFA of order (s,t) and vice versa and thereby establish their equivalence.

2. Construction of Regular Grammar from an SDVFA of order (s, t)

In this section, we will discuss how to construct regular grammar from an SDVFA of order (s, t).

Consider the an SDVFA of order $(1, 1)$ shown in Fig. 2.1(a). We construct a grammar that specifies the language accepted by the an SDVFA as follows:

1. Let $\{0, 1\}$, the set of input letters, be the set of terminals.
2. Let $\{A, B, C, D, E, T\}$, the set of states, be the set of nonterminals. (T is the trapping state or dead state).
3. Let A, the initial state, be the starting symbol.
4. Corresponding to transition $\delta(s_p, i_q) = s_k$ (s_p, s_k are states and i_q is the input symbol), there is a production $s_p \rightarrow i_q s_k$ if s_k is not an accepting state and there are two productions $s_p \rightarrow i_q s_k$ and $s_p \rightarrow i_q$ if s_k is an accepting state. For example, corresponding to the transition $\delta(D, 0) = E$, we have the productions $D \rightarrow 0E$ and $D \rightarrow 0$ and corresponding to the transition $\delta(C, 0) = D$, we have the production $C \rightarrow 0D$. We obtain the grammar shown in Fig. 2.1(b).

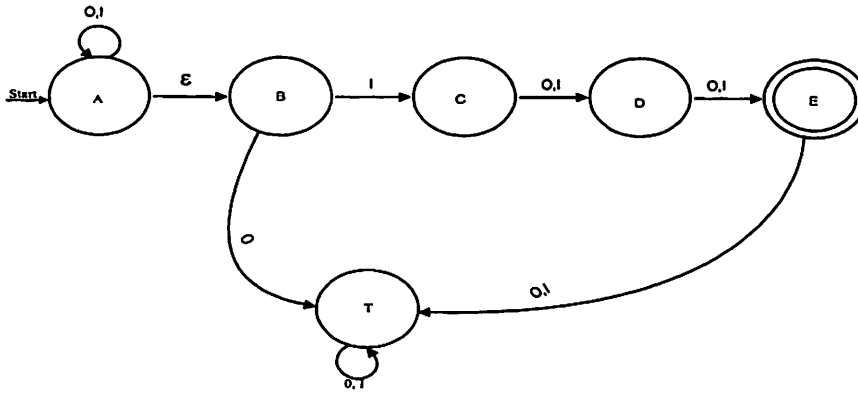

Fig. 2.1(a): SDVFA of order $(1, 1)$ that accepts all strings of 0's and 1's such that 3^{rd} symbol from the end is 1.

$$A \rightarrow 0A$$
$$A \rightarrow 1A$$
$$A \rightarrow \epsilon B$$
$$B \rightarrow 0T$$
$$B \rightarrow 1C$$
$$C \rightarrow 0D$$
$$C \rightarrow 1D$$
$$D \rightarrow 0E$$
$$D \rightarrow 0$$
$$D \rightarrow 1E$$
$$D \rightarrow 1$$
$$E \rightarrow 0T$$
$$E \rightarrow 1T$$
$$T \rightarrow 0T$$
$$T \rightarrow 1T$$

Fig. 2.1(b): Regular grammar representing the SDVFA of Fig. 2.1(a).

Observing that every nonterminal in the grammar represent a set of sequences that bring the SDVFA from the corresponding state to an accepting state, one would agree that the grammar in Fig. 2.1(b) specifies the language accepted by the SDVFA in Fig. 2.1(a).

3. Construction of an SDVFA from Regular Grammar

In this section, we will discuss the construction of a SDVFA of appropriate order from a given regular grammar.

Consider the regular grammar given in Fig. 3.1(a).

$$A \rightarrow 0A$$
$$A \rightarrow \epsilon B$$

$$B \rightarrow 0C$$
$$B \rightarrow 1D$$
$$C \rightarrow 0$$
$$C \rightarrow 1B$$
$$D \rightarrow 1$$
$$D \rightarrow 0A$$

Fig. 3.1(a)

We construct an SDVFA of order $(1,1)$ that accepts the sentences in the language specified by the grammar of Fig. 3.1(a) as follows:

1. Let {0,1}, the set of terminals, be the set of input letters.

2. Let there be a state corresponding to each nonterminal, with the state corresponding to the starting symbol being the starting state (state A in this case). Let there be an additional state E, which is an accepting state. Also, let there be an additional state T, which is called a **trapping state**. Whenever the SDVFA enters a trapping state, it will stay there and will not be able to go to any other state. Thus, if an input sequence brings the SDVFA to a trapping state, such an input sequence cannot possibly be a portion of a sentence in the language of the SDVFA.

3. For a production of the form $N_p \rightarrow i_q N_k$, there is a transition from state N_p to state N_k when the input is i_q (i_q may be real input or virtual input ϵ). For example, corresponding to the production $B \rightarrow 1D$, there is a transition from state B to state D when the input letter is 1. For a production of the form $N_p \rightarrow i_q$, there is transition from state N_p to the accepting state E when the input is i_q. For example, corresponding to the production $D \rightarrow 1$, there is a transition from state D to state E when the input letter is 1. On the other hand, if there is no production of the form $N_p \rightarrow I_q N_k$ or $N_p \rightarrow i_q$ for state N_p and input i_q, there is a transition from N_p to trapping state T when the input is i_q. For example, there is a transition from state A to state T when the input letter is 1. Finally,

for any input letter, there is a transition from accepting state E to trapping state T.

We thus obtain the SDVFA of order $(1,1)$ shown in Fig. 3.1(b). It is clear that the SDVFA we constructed accepts exactly the sentences in the language specified by the grammar in Fig. 3.1(a).

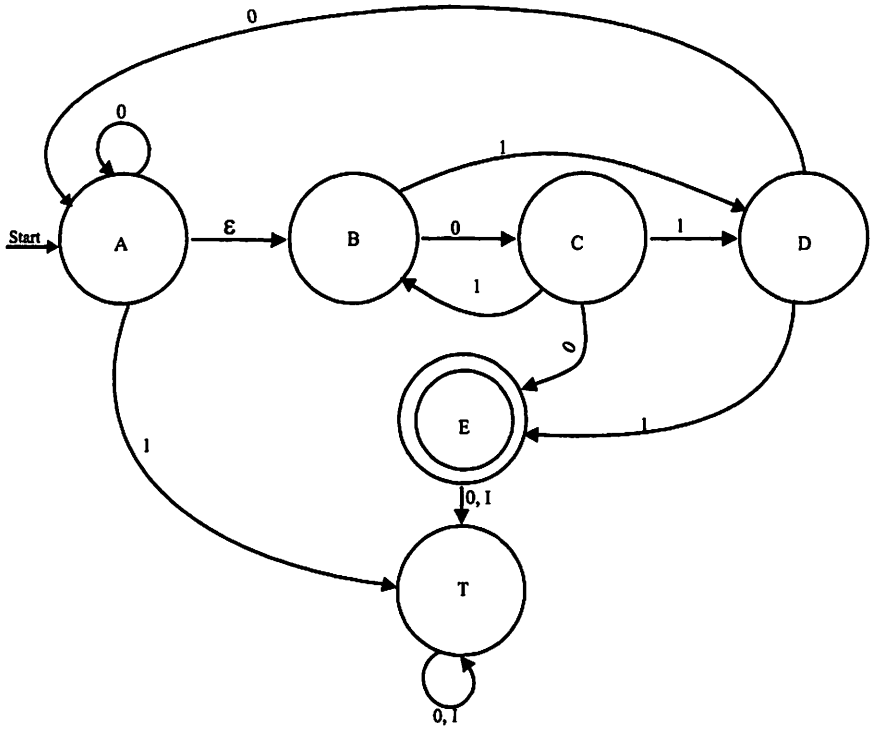

Fig. 3.1(b): SDVFA of order $(1,1)$ constructed from the grammar of Fig. 3.1(a).

It is clear that for every state N_p of SDVFA in Fig. 3.1(b), a transition on virtual input ϵ goes to the same state N_p if the transition is not shown explicitly.

We consider another example for the construction of an SDVFA of order $(2,0)$ from a regular grammar. Consider the grammar in 3.2(a).

$$A \rightarrow 0A$$
$$A \rightarrow 1B$$
$$B \rightarrow 0C$$
$$B \rightarrow 0D$$
$$C \rightarrow 0$$
$$C \rightarrow 1B$$
$$C \rightarrow 1D$$
$$D \rightarrow 1$$
$$D \rightarrow 0A$$

Fig. 3.2(a)

Note that $A \rightarrow \epsilon A$ is always true. We construct an SDVFA of order $(2,0)$ for the grammar of Fig. 3.2(a) by the procedure described in above discussion and is shown in Fig. 3.2(b). We are not showing ϵ-transitions in the transition diagram shown in Fig. 3.2(b) since the transitions from all states go to the same state.

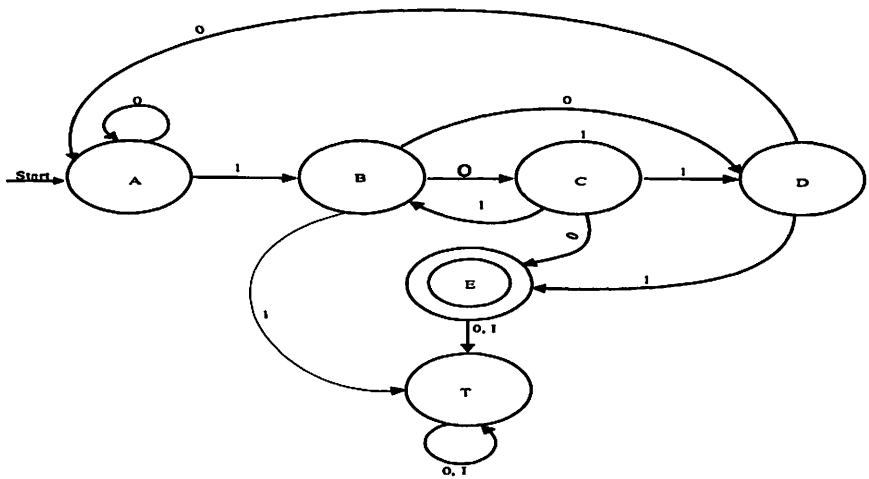

Fig. 3.2(b): SDVFA of order $(2,0)$ for the grammar of Fig. 3.2(a).

Thus, the languages specified by the grammars in Fig. 3.1(a) and Fig. 3.2(a) are indeed SDVFA languages.

4. Conclusion

From the discussions of Sections 2 and 3, we conclude that the class of SDVFA languages is exactly the class of Type-3 or regular languages.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

[2] A.V. Aho and J.D. Ullman, Foundations of Computer Science, Computer Science Press, New York, 1994.

[3] J. Anderson, Automata Theory with Modern Applications, Cambridge University Press, 2006.

[4] N. Chomsky, in review of Belevitch, V., *Language des machines et langage humain*, Language, 34 (1958), p.100.

[5] Chomsky, N., The logical Structure of Linguistic Theory, Manuscript, Harvard University, Plenum Press, New York and London, 1973.

[6] N. Chomsky, *Three models for the description of languages*, IRE Trans. on Information Theory, 2:3 (1956), pp.113-124.

[7] S. Eilenberg, *Automata, Languages and Machines*, Vol. A-B, Academic Press, New York, 1974.

[8] C.C. Elgot and G. Mezei, *On relations defined by generalized finite automata*, IBM J. of Res. and Dev., 9 (1965), 47-65.

[9] M. Ganesan, P. Bhattacharya and A. Jain, *A Notion of Virtual Deterministic Finite Automaton(VDFA) in Language Processing*, J. Comp. Math. Optim., 2 (2006), 181-206.

[10] A. Gill, Introduction to the Theory of Finite-State Machines, McGraw-Hill Book Company, New York, 1962.

[11] A. Jain, *Semi-deterministic virtual finite automaton(SDVFA) of order* (s,t), J. Comp. Math. Optim., 5 (2009), 1-22.

[12] A. Jain, *Equivalence of SDVFA of order* (s,t) *with DFA, VDFA, NFA and* ϵ*-NFA*, communicated.

[13] M.A. Harrison, Introduction to Formal Language Theory, Addison-Wesley Publishing Company, Reading Maas., 1978.

[14] F.C. Hennie, Finite-State Models for Logical Machines, John Wiley & Sons, New York, 1968.

[15] L.S. Levy, Discrete Structures of Computer Science, New Age International (P) Ltd., 2003.

[16] M.O. Rabin and D. Scott, *Finite automata and their decision problems*, IBM J. Research and Development 3:2 (1959), pp.115-125.

[17] M.P. Schutzenberger, *On the definition of a family of automata*, Information and Control, 4 (1961).

[18] C.E. Shannon and J. McCarthy, Automata Studies, Princeton Univ. Press, 1956.

32, Uttranchal
5, I.P. Extension
Delhi
India
E-mail: jainarihant@live.com