TWO WAY FINITE STATE GENERATORS 1)
Karel Culik II 2) and Emmerich Welzl 3)

INTRODUCTION

A problem for chain code picture languages posed by Maurer, Rozenberg, and Welzl [MRW], motivated the investigation of a device which we call nondeterministic two-way finite state generator. It has a writing head with a finite control and an (initially blank) two-way infinite working tape. At each step it writes a symbol into the current cell and moves in either direction. It has no reading capacity which means that that the action depends only on the current state of the finite control. Concerning the writing there are different interpretations some of them are the following: (1) the generator simply overwrites the current symbol under the writing head, (2) every cell contains an initially empty set of symbols (subset of a finite alphabet) and the generator adds a symbol to the set under the writing head (i.e. if the symbol is already in the set, the contents of the cell remains unchanged) and (3) the set of output symbols has a priority hierarchy and if the generator tries to write a symbol \( a \) of higher priority than the symbol \( b \) at the current position on the tape, then it overwrites this symbol \( b \) by \( a \); otherwise it leaves the symbol of higher priority on the tape.

In the next Section we give a formal definition of nondeterministic two-way finite state generators and the three mentioned output interpretations. Then we show for cases (2) and (3) above that the generated languages are regular, i.e. two-way finite state generators generate the same as one-way finite state generators (right-linear grammars, respectively). This corresponds to the result for acceptors, see Shepardson, [S] for a proof that two-way finite automata accept the same languages as one-way finite automata (a proof is also presented in Harrison, [H]). However, the authors do not see how to carry over the result for acceptors to generators, while the implication in the other direction is not too difficult to show.

DEFINITIONS

A two-way finite state generator is a tuple \( G = (Q, \delta, q_0, q_f, \Sigma, \text{INT}) \) where

(i) \( Q \) is a finite nonempty set of states

(ii) \( \delta: Q \times \Sigma \rightarrow 2^Q \) is a transition function, which gives the set of possible next states.

1) This work has been done during the first author's visit at the IIG.
2) Department of Computer Science, University of Waterloo, Waterloo, Canada.
3) Institutes for Information Processing, IIG, Technical University of Graz and Austrian Computer Society, Graz, Austria.
(iii) \( d : Q \to z^{\{-1,0,1\}} \) is a direction function, which indicates whether - being in a state \( q \in Q \) - the writing head can move to the right (1 \( \in d(q) \)), to the left (-1 \( \in d(q) \)) or can remain in the same position (0 \( \in d(q) \)) next.

(iv) \( q_0 \in Q \): an initial state

(v) \( q_f \in Q \): an accepting state

(vi) \( \Sigma \): a finite output alphabet and, finally,

(vii) \( \text{INT} : Q^+ \to \Sigma \) an interpretation function.

Using (i)-(v) we define the computations of \( G \), the language generated by \( G \) will then be defined using (vi) and (vii).

Computations of a generator \( G \) will be described by strings over \((Q \times Z)^1\), where \((q,k) \in Q \times Z\) stands for "current state: \( q \)" and "current position on the tape: \( k \)."

The move relations \( \vdash \) of the computation of a two-way finite state generator is defined as follows: Let \( c \in (Q \times Z)^* \), \((q,k) \in Q \times Z\). Then

\[
\begin{align*}
    c(q,k) &\vdash c(q,k)(q',k+i)
\end{align*}
\]

if \( q' \in \delta(q), i \in d(q) \). As usual, \( \vdash \) denotes the reflexive transitive closure of \( \vdash \).

The set of valid computations of \( G \) is defined as

\[
\text{comp}(G) = \{c \in (Q \times Z)^* | c=(q_0,0)c'(q_f,k) \text{ for some } c',k,(q_0,0) \vdash c\}
\]

For a valid computation \( c \in (Q \times Z)^+ \) the leftmost (rightmost) visited position by \( c \), shortly \( \text{lmb}(c) \) (\( \text{rm}(c) \)), is defined as the minimal (maximal) \( k \), such that \( c \) can be written as \( c=c_1(q,k)c_2 \) for some \( c_1,c_2 \in (Q \times Z)^* \), \((q,k) \in (Q \times Z)\).

---

Figure 1: Graphical display of a computation \( c \).

1) \( z \) denotes the set of integers.