SOFSEM'93 – Hrdoňov, Šumava, Czech Republic, 22.11. – 3.12.1993

A Remark about Forgetting Automata

František Mráz, Martin Plátek

Department of Computer Science, Charles University,
Malostranské nám. 25, 118 00 Praha 1, Czech Republic,
e-mail: mrazf@cspguk11.bitnet, platek@cspguk11.bitnet

Abstract: Forgetting automata are nondeterministic linear bounded automata whose rewriting capability is restricted as follows: each cell of the tape can only be "erased" (rewritten by a special symbol) or completely "deleted". We show that the "erasing" is in some sense more powerfull than the "deleting".

Key words: forgetting automaton, operation erase, operation delete, pushdown automaton

1 Introduction

This paper is a supplement of the contribution [JMP92b].

We deal with forgetting automata, which are nondeterministic linear bounded automata whose rewriting ability is restricted as follows: each cell of the tape can only be "erased" (rewritten by a special symbol) or completely "deleted".

We show here a simulation of nondeterministic forgetting automata with operations move to the right and erase operations by nondeterministic pushdown automata. By this simulation we complete the proof that the class of languages recognizable by this type of forgetting automata equals to the class of context-free languages (*CFL*).

The fact, that *CFL* is contained in the class of languages recognizable by this type of forgetting automata is an obvious consequence of the main result from [JMP92a].

The erase operation is more general than the delete operation. Delete operation can be simulated by erasing and skipping through erased item while preserving the current state. Moreover we show here that nondeterministic forgetting automata with operations move to the right and delete operations cannot recognize all context-free languages. It follows from this that the operation erase is strongly more powerfull than the delete operation in this context.

See [JMP93] for the detailed description of classes of languages recognizable by several types of forgetting automata.

2 Definitions

An *F-automaton* (forgetting automaton) F has a finite state control unit with one head moving on a linear (doubly linked) list of items (cells); each item contains a symbol from a finite alphabet. In the initial configuration, the control unit is in a fixed (initial) state, the list contains an input word bounded by special sentinels #, \$ and the head scans the item with the left sentinel #.

The computation of F is controlled by a finite set of instructions of the form $[q, a] \rightarrow [q_1, op]$, with the following meaning: according to the actual state q and the scanned symbol a, F may change the state to q_1 and perform op, one of the following six operations:

– MV_R, MV_L — moving the head one item to the right (left),

– ER_R, ER_L — erasing, i.e. rewriting the contents of the scanned item with a special symbol, say $*$ and moving the head one item to the right (left),

– DL_R, DL_L — deleting, i.e. removing the scanned item from the list and moving the head one item to the right (left)

Generally, F is *nondeterministic* (more than one instruction can be applicable at the same time).

An input *word is accepted by* F if there is a computation starting in the initial configuration which achieves a configuration with the control unit being in one of accepting states.

$L(F)$ denotes the language consisting of all words accepted by F; we say that F *recognizes* $L(F)$.

By $[O]$, where O is a subset of $\{MV_R, MV_L, ER_R, ER_L, DL_R, DL_L\}$, we denote the class of languages recognizable by F-automata using operations from O only. (We write $[Op_1, Op_2, \ldots, Op_n]$ instead of $[\{Op_1, Op_2, \ldots, Op_n\}]$).

The couple ER_R, ER_L we abbreviate by ER; similarly for DL and MV.

For the situations with the head scanning # (\$) we use the following technical assumption:

– on # only MV_R - instruction is applicable, and

– on \$ only move to the left-instruction is used (instead of ER_L or DL_L).

3 Results

First we show that forgetting automata with operations MV_R and ER only are not stronger than pushdown automata.

Theorem 1 *$[MV_R, ER]$ is a subset of CFL.*

The operation ER_R could be replaced by a sequence of operations ER_L, MV_R, MV_R. So it is easy to see, that $[MV_R, ER] = [MV_R, ER_L]$. Let F be a forgetting automaton with operations MV_R, ER_L. We will give an outline of a construction of a pushdown automaton M simulating F. F can operate on sequences of erased items. What can happen when F enters a sequence of n erased items in a state q from the left end:

- after some number of steps in the erased sequence F can halt in an accepting or non-accepting state,

- or after some number of steps F can leave the erased sequence through the left or the right end in a state q'.

F operates in a similar way when it enters a sequence of erased items from the right. We will call such description of operations of F (actually a set of functions) for some sequence u of n erased items for all states q of F a *behaviour of F on u* and denote it by B_n. In particular B_0 will denote the behaviour of F on the empty sequence of erased items (i.e. when F "enters" such a sequence from the left (right) in some state then F "leaves" it in the same state through the right (left) end).

There are only finitely many different behaviours of F on sequences of erased items. Having the behaviour B_n of F for some sequence of n erased items we can compute the behaviour B_{n+1} of F for the sequence of $n+1$ erased items without knowing the value n. Moreover having

behaviours B_1, B_2 of F for two sequences of erased items we can compute the behaviour B of F for the concatenation of these sequences of erased items.

The automaton M will simulate the automaton F in the following way: each contiguous sequence of erased items to the left from the scanning head of F is encoded in the pushdown of M as a behaviour. The only way how F can move to the left is the ER_L-operation. So there is at most one contiguous sequence of erased items to the right from the head of F. This sequence could be characterized by some behaviour B which will be stored in the finite control unit of M. In the case that there are no erased items to the right from the head of F the behaviour B equals to B_0.

Each (maximal) sequence s of steps of a computation of F in which only erased items are visited, except the last step of s when the segment of erased items is left, is simulated by one step of the pushdown automaton M. Each (maximal) contiguous sequence e of erased items will be represented by the corresponding behaviour. Using this behaviour M could nondeterministicaly guess in which state and through which end will F leave the sequence e or in which state F will halt without leaving e.

Let us describe how the remaining steps (which do not start on an erased items) are simulated:

a) MV_R instruction from an unerased item I with entering a state p:

   a1) $B = B_0$ (i.e. the item to the right from I will be visited for the first time) - M pushes the contents of the item I on the pushdown store and reads the next symbol from the input tape; B remains unchanged.

   a2) B is not B_0 (i.e. the item to the right from I was previously visited and consequently it is erased) - all consequent steps of F until it leaves the sequence of erased items are simulated nondeterministicaly using the behaviour B and the state q as described above. If F will halt in a state q_f then M halts in the state q_f. If F will leave the erased sequence to the left, then M enters a new state only. If F will leave the erased sequence to the right, then M pushes the contents of the item I on the pushdown store, pushes the behaviour B onto the pushdown store, reads the next symbol from the input tape, enters the new state and puts B_0 into B.

b) ER_L instruction on an unerased item I with entering a state q:

   Then the behaviour B stored in the finite control of M is changed to B' to represent the new longer sequence of erased items. The contents of the item I is poped from the pushdown store and the top of the pushdown store is inspected. We distinguish two cases.

   b1) There is an input symbol on the top of the pushdown store. Then M enters the state q only.

   b2) There is a behaviour B'' on the top of the pushdown store - i.e. the automaton F enters a sequence of erased items from the right. Then M computes a behaviour B_c representing the whole sequence of erased items on the working list of F (using B' and B'') and M nondeterministicaly chooses in which state and in which direction will the new erased segment be left. If F will halt in a state q_f then M halts in the state q_f. When F will leave the erased segment to the left then M pops B'' from the pushdown store, stores the new behaviour B_c in B and enters the chosen state. When F will leave the erased sequence to the right, then M changes the behaviour on the top of the pushdown store to the new one, reads the next input symbol from the input tape and pushes it onto the top of the pushdown store and enters the chosen state.

It is easy to see that if there is an accepting computation of the F-automaton F on a word w, then there is an accepting computation of the pushdown automaton M on w. On the other side, when there is no accepting computation of F on an input word w, then M cannot accept it. So $L(F) = L(M)$.

The following theorem can be easily derived from the main result from [JMP92a] and from the previous theorem.

Theorem 2 $[MV_R, ER]$ *is equal to the* CFL.

To show that the operation *ERASE* is more powerful than the operation *DELETE* we will show that by replacing the *ERASE* operation by the operation *DELETE* in the above mentioned class of forgetting automata we get a subclass of *CFL*.

Theorem 3 $[MV_R, DL]$ *is a strict subclass of* CFL.

The inclusion $[MV_R, DL] \subset CFL$ follows trivially from Theorem 1. This inclusion is proper and it could be proved using the language $L(G)$ generated by the following context-free grammar $G = (\{S, A_1, A_2\}, \{a_1, a_2, d_1, d_2, s\}, S, R)$ where S is the starting nonterminal and R:

$$\begin{aligned} S &\rightarrow A_1 S d_1 \mid A_2 S d_2 \mid s \\ A_1 &\rightarrow c A_1 c \mid a_1 \\ A_2 &\rightarrow c A_2 c \mid a_2 \end{aligned}$$

Obviously $[MV_R, DL] = [MV_R, DL_L]$ (similarly as in the proof of Theorem 1). The language $L(G)$ cannot be recognized by a forgetting automaton with operations MV_R and DL_L only.

The complete proof is too technical and rather long for presentation in this proceedings and can be found in [JMP93]. It may be interesting, that in the proof there are used two notions, dependency and projectivity, which we have learnt from linguistics. These are the fundamental properties of "moving trees" introduced in [JMP93]. For each computation of a forgetting automaton with operations MV_R and DL_L only we can construct a moving tree which comprises the complete information about the computation. For these trees we can prove two "pumping lemmas". The proof in [JMP93] is based on these two pumping lemmas and a careful analysis of some sets of moving trees for accepting computations of F-automata with operations MV_R and DL_L only.

4 Conclusions

Obviously $[ER] = [DL] = [MV_L, ER] = [MV_L, DL]$ (see [JMP92b] or [JMP93]).

We have shown that the operation *ERASE* is more powerfull than *DELETE* when combined with the operation MV_R. I.e. $[MV_R, DL]$ is a proper subset of $[MV_R, ER]$.

In [JMP92b] and [JMP93] we conjectured that $[MV, DL]$ is a proper subset of $[MV, ER]$ but this is still an open problem.

5 References

[JMP92a] Jančar P., Mráz F., Plátek M.: *Characterization of Context-Free Languages by Erasing Automata,* in Proceedings of the 17th International Symposium on Mathematical Foundations of Computer Science 1992, Lecture Notes in Computer Science, Vol. 629, Springer-Verlag, Berlin 1992, pp.305–314

[JMP92b] Jančar P., Mráz F., Plátek M.: *Forgetting automata and the Chomsky hierarchy*, in Proc. SOFSEM '92, Ždiar, Slovakia, November 1992, pp. 41-44

[JMP93] Jančar P., Mráz F., Plátek M.: *A Taxonomy of Forgetting Automata,* Tech. Report No. 101, KTI MFF Charles University, Prague, June 1993