Linguistic Applications of Finite Automata with Translucent Letters

Benedek Nagy[1] and László Kovács[2]
[1]*Faculty of Informatics, University of Debrecen, PO Box 12, Debrecen, Hungary*
[2]*Department of Information Technology, University of Miskolc, Miskolc-Egyetemváros, Hungary*

Keywords: Finite Automata, Mildly Context-sensitive Languages, Natural Languages, Formal Linguistics, Free-order Languages.

Abstract: Finite automata with translucent letters do not read their input strictly from left to right as traditional finite automata, but for each internal state of such a device, certain letters are translucent, that is, in this state the automaton cannot see them. We address the word problem of these automata, both in the deterministic and in the nondeterministic case. Some interesting examples from the formal language theory and from a segment of the Hungarian language is shown using automata with translucent letters.

1 INTRODUCTION

The finite automaton is a fundamental computing device for accepting languages. Its deterministic version (DFA) and its nondeterministic version (NFA) both accept exactly the regular languages, and they are being used in many areas like compiler construction, text editors, computational linguistics, etc. For regular languages the word problem is decidable by a real-time (i.e., linear) computation. However, the expressiveness of regular languages is quite limited, and thus, these automata are too weak for several further applications. Accordingly, much more powerful models of automata have been introduced and studied like, e.g., pushdown automata, linear-bounded automata, and Turing machines. But this larger expressive power comes at a price that certain algorithmic questions like the word problem or the emptiness problem become more complex or even undecidable. Hence, when dealing with applications, for example in natural language processing or concurrency control, it is of importance to find models of automata that reconcile two contrasting goals: they have sufficient expressiveness and, at the same time, a moderate degree of complexity.

An interesting question in application of formal languages is whether the sentences of natural languages (NL) can be modeled with a class of the Chomsky hierarchy or not. There are many different views and approaches in the literature regarding the position of natural languages in this hierarchy. It is folklore that finite languages are regular, and hence the simplest model for NL uses regular form approach based on the idea that the set of all sentences of humans are finite (there is only a finite number of people live/lived and each of them said/wrote only a finite number of sentences). There are also different approaches which use context-free grammars (Gazdar, 1982) while others like Matthews (Matthews, 1979) consider NL even more complex than the recursively enumerable languages. Since we do not understand all features of the human brain, one may believe that human brain can do more complex 'computations' than the computers/Turing machine can do.

The first finite state model for NL was developed in 1955 by Hockett (Hockett, 1955). Later, in 1969, Reich (Reich, 1969) has argued that NLs are not self-embedding to any arbitrary degree. Another argument for regularity is given by among others Sullivan (Sullivan, 1980): an individual neuron in the brain can be modeled with a finite automaton and the brain contains only finite number of neurons thus the resulting structure also corresponds to a finite automaton. The number of states of the resulting non-deterministic finite automaton is approximated with 10^{10^9}. Thus the only way to prove that NL are more complex than regular grammars is to exhibit some infinite sequences of grammatical sentences (Kornai, 1985). Chomsky (Chomsky, 1957) himself argues that English and other NLs are not regular languages. The most famous example for non-regular behaviour is the following pattern (Wintner, 2002):

Nagy B. and Kovács L..
Linguistic Applications of Finite Automata with Translucent Letters.
DOI: 10.5220/0004359504610469
In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence* (LAFLang-2013), pages 461-469
ISBN: 978-989-8565-38-9

A white male $(whom\ a\ white\ male)^n$ $(hired)^n$ hired another white male.

Also in the case, when the argumentation is accepted that the given structure can't be realized in infinite depth, it is clear that the corresponding base regular grammar/finite automata for finite depth may have a very large complexity. Thus, from a practical point of view it is worth to find more compact grammars that can describe the special behaviour of the different natural languages.

There are also very strong arguments that NLs are not context-free. In the area of formal languages in conjunction with computational linguistics and natural language processing the formulation of the notion of "mildly context-sensitive languages" (Joshi, 2010; Mery et al., 2006) has been appeared and used. These classes are proper subclasses of the class of context-sensitive languages and proper superclasses of the class of context-free languages. They contain some typical examples of non-context-free languages that show important features of natural languages, e.g., $\{a^n b^n c^n | n \geq 0\}$, $\{ww | w \in \{a,b\}^*\}$ and $\{a^n b^m c^n d^m | m,n \geq 0\}$. At the same time mildly context-sensitive languages share many of the nice properties with the context-free languages. For example, they have semi-linear Parikh images and their parsing complexity is polynomially bounded.

An interesting extension of the class of finite automata that vastly increases its expressive power, the so-called finite automaton (or finite-state acceptor) with translucent letters (*NFAwtl* for short) was introduced in (Nagy and Otto, 2011). The idea of this new model comes from the research of cooperative distributed systems of stateless deterministic restarting automata with window size 1 (Nagy and Otto, 2012a; Nagy and Otto, 2012b). An NFAwtl does not read its input strictly from left to right as the traditional finite automaton does, but for each of its internal states, certain letters are translucent, that is, in this state the NFAwtl cannot see them. Accordingly, it may read (and erase) a letter from the middle or the end of the given input. They accept certain non-regular and even some non-context-free languages, but all languages accepted by NFAwtls have semi-linear Parikh images (Nagy and Otto, 2012a). These issues are important for the linguistic applications point of view. In contrast to the classical finite-state acceptors, the deterministic variants of the NFAwtls, the so-called *DFAwtls*, are less expressive than the nondeterministic ones (Nagy and Otto, 2012b). In this paper, as a continuation of the work started in (Nagy and Otto, 2011), we consider the word problem of these automata by showing a nondeterminsitic/deterministic linear-time algorithm that decides if a given word is accepted or not by an NFAwtl/DFAwtl, respectively. Some linguistical examples are also shown to demonstrate the efficiency of NFAwtl/DFAwtl.

This paper is structured as follows. In Section 2 we recall the definition of the finite automata with translucent letters and we also present an example. In Section 3 the word problem (parsing) is considered with some further examples, while in Section 4 we show examples for the usage of finite automata with translucent letters modeling some phenomena of the Hungarian language. In our investigation, the Hungarian language was selected which differs from English in the following important aspects:

- it is an agglutinative language (in Hungarian most grammatical information is given through suffixes: cases, conjugation, etc.),
- it has no dominant word order (as we will see, in Hungarian, several kinds of order of the words of a sentence can be considered emphasizing slightly different special meaning),
- reduced use of postpositions (since suffixes are often equivalent to English prepositions, there are only few postpositions in Hungarian).

In Section 5 we summarize our work and give some open problems for future work.

2 FINITE-STATE ACCEPTORS WITH TRANSLUCENT LETTERS

In this section we fix our notation and recall the definition and some basic facts about the finite automata with translucent letters.

A *nondeterministic finite automaton* (NFA) is described by a tuple $A = (Q, \Sigma, I, F, \delta)$, where Q is a finite set of internal states, Σ is a finite alphabet of input letters, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \to 2^Q$ is a transition relation. If $|I| = 1$ and $|\delta(q,a)| \leq 1$ holds for all $q \in Q$ and all $a \in \Sigma$, then A is a *deterministic finite automaton* (DFA).

An NFA A works as follows. It is given an input string $w \in \Sigma^*$, and A starts its computation/run in a state q_0 that is chosen nondeterministically from the set I of all initial states. This configuration is encoded as $q_0 w$. Now it reads the first letter of w, say a, thereby deleting this letter, and it changes its internal state to a state q_1 that is chosen nondeterministically from the set $\delta(q_0, a)$. Should $\delta(q_0, a)$ be empty, then A gets stuck (in this run), otherwise, it continues its run by

reading letters until w has been consumed completely. We say that A accepts w with a run if A is in a final state $q_f \in F$ after reading w completely at the end of this run. By $L(A)$ we denote the set of all strings $w \in \Sigma^*$ for which A has an accepting computation in the sense described above.

It is well-known that the class $\mathcal{L}(\mathsf{NFAwtl})$ of languages $L(A)$ that are accepted by NFAs coincides with the class REG of regular languages, and that DFAs accept exactly the same languages.

Now we recall a variant of the nondeterministic finite automata that does not process its input strictly from left to right (Nagy and Otto, 2011).

Definition 1. *A* finite-state acceptor with translucent letters *(NFAwtl) is defined as a 7-tuple* $A = (Q, \Sigma, \$, \tau, I, F, \delta)$, *where* Q *is a finite set of internal states,* Σ *is a finite alphabet of input letters,* $\$ \notin \Sigma$ *is a special symbol that is used as an* endmarker, $\tau : Q \to 2^\Sigma$ *is a* translucency mapping, $I \subseteq Q$ *is a set of initial states,* $F \subseteq Q$ *is a set of final states, and* $\delta : Q \times \Sigma \to 2^Q$ *is a* transition relation. *For each state* $q \in Q$, *the letters from the set* $\tau(q)$ *are translucent for* q, *that is, in state* q *the automaton* A *does not see these letters. A is called* deterministic, *abbreviated as DFAwtl, if* $|I| = 1$ *and if* $|\delta(q,a)| \leq 1$ *for all* $q \in Q$ *and all* $a \in \Sigma$.

An NFAwtl $A = (Q, \Sigma, \$, \tau, I, F, \delta)$ works as follows. For an input word $w \in \Sigma^*$, it starts in a nondeterministically chosen initial state $q \in I$ with the word $w \cdot \$$ on its input tape. A single step computation of A is as follows. Assume that $w = a_1 a_2 \cdots a_n$ for some $n \geq 1$ and $a_1, \ldots, a_n \in \Sigma$. Then A looks for the first occurrence from the left of a letter that is not translucent for the current state q, that is, if $w = uav$ such that $u \in (\tau(q))^*$ and $a \notin \tau(q)$, then A nondeterministically chooses a state $q' \in \delta(q,a)$, erases the letter a from the tape thus producing the tape contents $uv \cdot \$$, and its internal state is set to q'. In state q' the automaton considers the tape $uv\$$ and continues the process by another single step computation looking for the first visible letter of uv at state q'. In case $\delta(q,a) = \emptyset$, A halts without accepting. Finally, if $w \in (\tau(q))^*$, then A reaches the \$-symbol and the computation halts. In this case A accepts if q is a final state; otherwise, it does not accept. Observe that this definition also applies to configurations of the form $q \cdot \$$, that is, $q \cdot \varepsilon \cdot \$ \vdash_A \mathsf{Accept}$ holds if and only if q is a final state. A word $w \in \Sigma^*$ is *accepted by A* if there exists an initial state $q_0 \in I$ and a computation $q_0 w \cdot \$ \vdash_A^* \mathsf{Accept}$, where $\vdash_A^*$ denotes the reflexive transitive closure of the single-step computation relation $\vdash_A$. Now $L(A) = \{ w \in \Sigma^* \mid w \text{ is accepted by } A \}$ is the *language accepted by A*.

The classical *nondeterministic finite automata* (NFA) is obtained from the NFAwtl by removing the endmarker \$ and by ignoring the translucency relation τ, and the *deterministic finite-state acceptor* (DFA) is obtained from the DFAwtl in the same way. Thus, the NFA (DFA) can be interpreted as a special type of NFAwtl (DFAwtl). Accordingly, all regular languages are accepted by DFAwtl. Moreover, DFAwtls are much more expressive than standard DFAs as shown by the following example.

Example 1. Let $A = (Q, \Sigma, \$, \tau, I, F, \delta)$, where $Q = \{q_0, q_1, q_2\}$, $I = \{q_0\} = F$, $\Sigma = \{a, b, c, d\}$, and the functions τ and δ are defined as follows:

$$\begin{array}{lcllcl} \tau(q_0) & = & \emptyset, & \delta(q_0,a) & = & \{q_1\}, \\ & & & \delta(q_0,b) & = & \{q_2\}, \\ \tau(q_1) & = & \{a,b\}, & \delta(q_1,c) & = & \{q_0\}, \\ \tau(q_2) & = & \{b\}, & \delta(q_2,d) & = & \{q_0\}, \end{array}$$

and $\delta(q,x) = \emptyset$ for all other pairs $(q,x) \in Q \times \Sigma$. Observe that A is in fact a DFAwtl.

It can be shown that $L(A)$ consists only some of words with $|w|_a = |w|_c$ and $|w|_b = |w|_d$, moreover $L(A) \cap (a^* \cdot b^* \cdot c^* \cdot d^*) = \{ a^n b^m c^n d^m \mid n, m \geq 0 \}$ and thus this language is not context-free.

It is shown that already DFAwtls accept non-context-free languages.

An NFAwtl $A = (Q, \Sigma, \$, \tau, I, F, \delta)$ is described more transparently by a graph, similar to the graph representation of standard NFAs. A state $q \in Q$ is represented by a node labelled with q, where the node of an initial state p is marked by a special incoming edge without a label, and the node of a final state p is marked by a special outgoing edge with label $(\tau(p))^*$. For each state $q \in Q$ and each letter $a \in \Sigma \setminus \tau(q)$, if $\delta(q,a) = \{q_1, \ldots, q_s\}$, then there is a directed edge labelled $((\tau(q))^*, a)$ from the node corresponding to state q to the node corresponding to state q_i for each $i = 1, \ldots, s$. The graph representation of the DFAwtl A of Example 1 is given in Figure 1. (Using the notation $\emptyset^* = \{\varepsilon\}$.)

According to the definition, an NFAwtl may accept a word without processing it completely. This, however, is only a convenience that makes for simple instructions, since for every NFAwtl A one can effectively construct an NFAwtl B such that $L(B) = L(A)$, but for each word $w \in L(B)$, each accepting computation of B on input w consists of $|w|$ many reading steps plus a final step that accepts the empty word (i.e., the input is totally processed). Nevertheless it is an open problem whether the same fact holds for DFAwtls in general. If A is an NFAwtl on Σ that is accepting only totally processed input, then by removing the translucency relation from A, we obtain a standard NFA A' that accepts a letter equivalent language of the original language $L(A)$, moreover $L(A) \supseteq L(A')$.

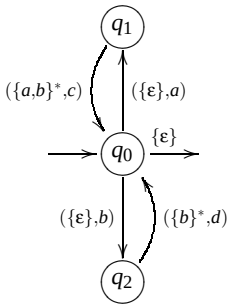

Figure 1: The graphical representation of the DFAwtl A of Example 1.

a_1		a_1	a_1		...		a_1
a_2	a_2				...	a_2	
:					...		
a_n				a_n	...		

Figure 2: Divided tape model of NFAwtl.

3 THE PARSING PROBLEM

In this section we address the word problem, i.e., how we can decide if a given word is accepted by an NFAwtl/DFAwtl.

One may feel that in every step of our automaton the input should be processed from the beginning by searching an occurrence of the letter indicated by the transition; and in this way the time complexity of the word problem looks quadratic. In the next part of this section we show a representation method which gives lower complexity for the word problem.

Let us start this section with a kind of description of our automata, i.e., how it could work. Our aim is to reach easily the next occurrence of a given letter. To do so, first, let us divide the tape into as many parts as the cardinality of the alphabet (about a similar way as a one-tape Turing machine can simulate a multi-tape Turing machine (Hopcroft and Ullman, 1969)). See Figure 2 as well. In each division exactly one of the letters is used: there is a bijection from Σ to the parts of the tape.

Then in each state those parts of the tape are not used that are assigned to translucent letters. With this picture in mind, one can construct the 'linked-list' data structure of the tape content in a linear number of steps (by the length of the input). The notation w_n is used for the n-th letter of the word w. The linked-list structure contains the array $STORE$ with dimension $|w|$ and $|\Sigma|$ pointers. A pointer $HEAD_a$ shows the first occurrence of letter a in w. In that position in the array there is a value that shows the second occurrence of letter a in w, etc. At the last occurrence of a there is a special marker that shows that there is no further occurrences of the letter a. The formal algorithm is shown in Figure 3. Starting from the end of the word we easily find the last occurrence of every type of letters of the word, we mark these places of $STORE$ by special markers ($NULL$). Further preprocessing the input we put every place to $STORE$ the value that indicates the next occurrence of the given letter. Finally, the pointers $HEAD_j$ show the first occurrences of the letters.

Then the constructed data structure is used in computation as a representation of the input word (see Figure 4). By the constructed data structure one can easily access the first occurrence of any letters a of any word w by the pointer $HEAD_a$. When a machine should read a letter a then one needs a constant number (at most $|\Sigma|-1$) of comparisons: if $HEAD_a > HEAD_b$ for any non-translucent letters b occurring in the unprocessed part of the input (here the value $NULL$ is considered as ∞), then the machine gets stuck, else a is read by the machine: it is done by erasing the current head of the list representing a's: let $HEAD_a = STORE_{HEAD_a}$. When all lists are empty, i.e., $HEAD_a = NULL$ for every input letter a, then the input is fully processed.

In this way a DFAwtl processes the input in linear time: linear preprocessing and linear processing (the number of simple operations (comparisons and assignment statements) are bounded by $(|\Sigma|+2|w|)+(|\Sigma|-1)|w|)$).

Therefore the word problem for DFAwtl is almost as simple as for DFA, this family of languages has this very pleasant and effective property. (Note that here we allowed to compare numbers in a fixed time complexity, however the stored numbers of the array $STORE$ depend on the length of the input.)

For NFAwtl the word problem using our representation is linear with a nondeterministic version of our algorithm (i.e., the problem is in the class $NLIN$).

Further in this section we present some additional examples. The next example is one of the most essential context-free languages.

Example 2. The Dyck language is accepted by the DFAwtl shown in Figure 5. The input *abaababb* is represented by $HEAD_a = 1$, $HEAD_b = 2$, $STORE = 35467\infty8\infty$ (where ∞ represents the value $NULL$ as discussed before). The run of the machine on this input is as follows:

State q_0, $HEAD_a = 1$ ($HEAD_a < HEAD_b$) is changing to $STORE_{HEAD_a} = STORE_1 = 3$; the next state is q_1.

State q_1, $HEAD_b = 2$ is changing to $STORE_{HEAD_b} = STORE_2 = 5$; the next state is

```
Input:  the alphabet Σ and the input word w.
Output:  linked lists for every element a ∈ Σ:
    stored in a |w|-long string STORE and HEADs of the lists.
  Initialization:  For every a ∈ Σ let HEAD_a = NULL.
  For i = |w| downto 1 step −1 do
      Let STORE_i = HEAD_{w_i}
      Let HEAD_{w_i} = i
   EndFor.   [....]
```

Figure 3: Preprocessing algorithm to obtain linked list representation of the input word.

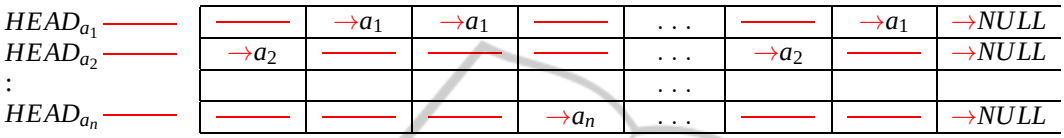

Figure 4: Linked list model of NFAwtl with preprocessed input.

q_0.

State q_0, $HEAD_a = 3$ ($HEAD_a < HEAD_b$) is changing to $STORE_{HEAD_a} = STORE_3 = 4$; the next state is q_1.

State q_1, $HEAD_b = 5$ is changing to $STORE_{HEAD_b} = STORE_5 = 7$; the next state is q_0.

State q_0, $HEAD_a = 4$ ($HEAD_a < HEAD_b$) is changing to $STORE_{HEAD_a} = STORE_4 = 6$; the next state is q_1.

State q_1, $HEAD_b = 7$ is changing to $STORE_{HEAD_b} = STORE_7 = 8$; the next state is q_0.

State q_0, $HEAD_a = 6$ ($HEAD_a < HEAD_b$) is changing to $STORE_{HEAD_a} = STORE_6 = \infty$; the next state is q_1.

State q_1, $HEAD_b = 8$ is changing to $STORE_{HEAD_b} = STORE_8 = \infty$; the next state is q_0.

The input is empty (both $HEAD_a$ and $HEAD_b$ are $NULL$ and q_0 is a final state, the input is accepted.

The input *abbabaab* is represented by $HEAD_a = 1$, $HEAD_b = 2$, $STORE = 435687\infty\infty$. The run of the machine on this input is as follows:

State q_0, $HEAD_a = 1$ ($HEAD_a < HEAD_b$) is changing to $STORE_{HEAD_a} = STORE_1 = 4$; the next state is q_1.

State q_1, $HEAD_b = 2$ is changing to $STORE_{HEAD_b} = STORE_2 = 3$; the next state is q_0. The first two letters of the input are already processed.

State q_0, $HEAD_a = 4$, $HEAD_b = 3$, therefore $HEAD_a > HEAD_b$ this transition cannot be executed. The machine gets stuck. This input is not accepted, *abbabaab* is not in the Dyck language.

Further, we present two non-context-free languages that are closely related to basic mildly context-sensitive languages.

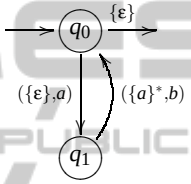

Figure 5: The DFAwtl of Example 2.

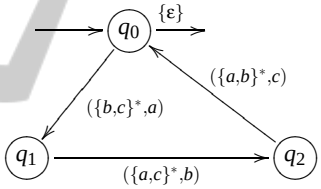

Figure 6: The DFAwtl of Example 3.

Example 3. Let $A = (Q, \Sigma, \$, \tau, I, F, \delta)$, where $Q = \{q_0, q_1, q_2\}$, $I = \{q_0\} = F$, $\Sigma = \{a, b, c\}$, and the functions τ and δ are defined as follows:

$$\begin{array}{lllllll} \tau(q_0) & = & \{b,c\}, & & \delta(q_0,a) & = & \{q_1\}, \\ \tau(q_1) & = & \{a,c\}, & & \delta(q_1,b) & = & \{q_2\}, \\ \tau(q_2) & = & \{a,b\}, & & \delta(q_2,c) & = & \{q_0\}, \end{array}$$

and $\delta(q,x) = \emptyset$ for all other pairs $(q,x) \in Q \times \Sigma$. The language $\{w \in \{a,b,c\}^* || w|_a = |w|_b = |w|_c\}$ is accepted by this DFAwtl. Its graphical representation is shown in Figure 6.

An example run on input *abbacbcca* is :

Representation of the input: $HEAD_a = 1$, $HEAD_b = 2$, $HEAD_c = 5$, $STORE = 43697\infty8\infty\infty$.

Starting from the initial state q_0 an a is read by changing $HEAD_a$ from 1 to $STORE_1 = 4$.

Then in state q_1 a b is read by increasing $HEAD_b$ from 2 to $STORE_2 = 3$.

In state q_2 a c is read by changing $HEAD_c$ from 5 to $STORE_5 = 7$.

Now the system is in q_0 and an a is read: $HEAD_a$ is changed to $STORE_4 = 9$.

In state q_1 $HEAD_b$ is increased to $STORE_3 = 6$.

In state q_2 $HEAD_c$ is changed to $STORE_7 = 8$.

In state q_0 $HEAD_a$ is increased to $STORE_9 = \infty$.

In state q_1 $HEAD_b$ is increased to $STORE_6 = \infty$.

In state q_2 $HEAD_c$ is changed to $STORE_8 = \infty$ and the system is arrived to q_0.

Since all lists are empty (all $HEAD$s are $NULL$) the input is fully processed and the system is in its final state, the input is accepted.

Note that in this example all letters that are not being read in a step are translucent, therefore the run does not need any comparisons. The machine may get stuck only if one of the letters is running out...

The language of the previous example intersected by the regular language $a^*b^*c^*$ gives the language $a^nb^nc^n$. The language $\{a^nb^mc^nd^m\}$ can be recognized in a similar way: it can be obtained as an intersection of the DFAwtl language presented in Example 1 and the regular language $a^*b^*c^*d^*$. These languages are belonging to mildly context-sensitive language families and they are important from linguistic point of view. The next language is closely connected to the copy language that is also belonging to mildly context-sensitive language families.

Example 4. The disjoint copy language $\{ww'|w \in \{a,b\}^*, w' \in \{a',b'\}, w' = h(w)$, where the alphabetic morphism h maps the letters to their primed versions $\}$ is accepted in the following way. Consider the DFAwtl shown in Figure 7. The language accepted by the DFAwtl of Figure 7 intersected by the regular language $(a+b)^*(a'+b')^*$ gives exactly the disjoint copy language.

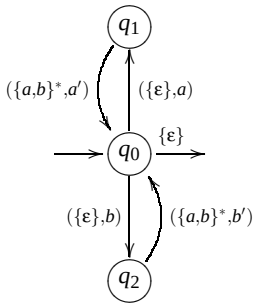

Figure 7: The graphical representation of the DFAwtl of Example 4.

4 APPLICATIONS IN A NATURAL LANGUAGE: MODELING SOME STRUCTURES OF HUNGARIAN LANGUAGE

In the previous section we gave some examples how NFAwtl's/DFAwtl's can be used modelling languages that are not context-free and closely connected to the main mildly context-sensitive languages. In this section we make a further step: we use NFAwtl's to present some features of a natural language, namely, of the Hungarian language.

In our investigation, the Hungarian language was selected which differs from English in many aspects. As we already mentioned, the Hungarian language is an agglutinative language, it has no dominant word order, and there is a reduced use of postpositions.

Our next example is modeling the two types of conjugation of the Hungarian language. The Hungarian language is a free order language therefore our new model can effectively be used. In our example we consider a very small segment of the language focusing on the phenomenon.

Example 5. The used 'alphabet', i.e., the Hungarian words can be seen in Tables 1 and 2. Conjugation of verbs can be seen in Table 2.

One of the most widely investigated distinguishing features of languages is the ordering of subject (S), object (O) and verb (V) within a sentence. Theoretically there are seven different ways and each way is represented by a set of living languages. According to the statistical analysis (Dryera et al., 2012), the dominating sequence is the SOV order with about 565 languages, the smallest group is the cluster with 4 languages for OSV order. An example of OSV order can be found in the Nadeb language, where the sentence 'the child sees the jaguar' is given with

awad (jaguar) kalapéé (child) hapúh (to see)

(Note that in Hungarian it also make sense to use OSV order: Jaguárt a gyerek lát. (a gyerek = the child, lát = (can) see) However, in Hungarian this sentence may have an additional meaning, it underlines the child, i.e., not the adult (or anybody else), but the child sees the jaguar.

In a significant number of languages, no single dominant order can be found. This languages use a relatively free word order. According to some recent approaches (Dryer, 1997) the SOV order has a marginal role in categorization of the languages as in more languages some of these components can be eliminated from the sentences, the corresponding clause will be pronominal or it is expressed by some verbal affixes. It is argued that a more useful typology is one based on two more basic features, whether the language is OV or VO and whether it is SV or VS.

Table 1: Hungarian words used in Example 5.

person	én *I*	te *you*	ő *she/he*	Jani *Johnny*	Mari *Mary*
thing	a könyv	az újság	a kenyér	a keksz	a sajt
	the book	*the newspaper*	*the bread*	*the biscuit*	*the cheese*
object	a könyvet	az újságot	a kenyeret	a kekszet	a sajtot

Table 2: Hungarian verbs used in Example 5.

	(I)	*(you)*	*(she/he/it)*	*(I)*	*(you)*	*(she/he/it)*
definite	eszem	eszed	eszi	olvasom	olvasod	olvassa
	eat			*read*		
indefinite	eszek	eszel	eszik	olvasok	olvasol	olvas
situation	fekszem	fekszel	fekszik	vagyok *am*	vagy *are*	van *is*
	lay			*be*		

Correct sentences in Hungarian can be formed in the following way: a person or a thing and a situation verb can be paired in any order (respecting the person of the conjugation), e.g., “Én fekszem.”, “Vagyok én.”, “Fekszel te.”, “Te vagy.”, “Ő fekszik.”, “Fekszik Jani.”, “Mari van.”, “Az újság van.”, “A könyv fekszik.”

A person and a verb in indefinite form can also be paired in any order (respecting the person of the conjugation), e.g., “Én eszek.”, “Olvasok én.”, “Te eszel.”, “Olvas ő.”, “Mari eszik.”, “Eszik Jani.”

A person, a verb in definite form and an object can also be grouped to form a sentence in any order (respecting the person of the conjugation), e.g., “Én eszem a kenyeret.”, “A kenyeret te eszed.”, “Eszi Mari a kenyeret.”, “Én az újságot olvasom.”, “Te a könyvet olvasod.”, “A kekszet eszi Jani.”, “Én eszem kenyeret.”, “Olvassa ő a könyvet.”

Therefore the automaton may look first for the subject (having everything else translucent), and then depending on the object the automaton looks for the verb (with everything else translucent). If definite version of a verb is used with a person, then the automaton will check the existence of the object also.

From the theoretical viewpoint, the main interest in modeling natural language grammars focuses on the sentences with unlimited length. Regarding the set of sublanguages belonging to the class accepted by NFAwtl the following condition should be met (Nagy and Otto, 2011): any language accepted by an NFAwtl should contain a regular sublanguage being letter equivalent with the language itself. Allowing an unbounded length, this regular sublanguage must have the following structure

$$S_1 S_2^* S_3$$

where S_1 ,S_2 ,S_3 are of finite length. Thus the pattern accepted by the NFAwtl is letter equivalent with this pattern. In the simplest cases, this part is equal to

- the repetition of a fixed subsequence or
- the repetition of any permutations of elements of the subsequence or
- the permutation of equal numbers of elements from each different symbols in the subsequence

Considering the first case, the pattern includes the repetition of the same element:

$$S_1 w^* S_3$$

A sample sentence can be given as

Péter szereti Annát, Évát, Katit, Marikát,... (Peter likes Anna, Eve, Kate, Mary,...)

This kind of pattern can be described with a simple regular grammar, thus no translucent symbols are needed. Thus, the power of *NFAwtl* can be demonstrated at such pattern where the ordering of the elements can be arbitrary. The next sentence demonstrates this kind of pattern:

Anna egy könyvet olvas arról, hogy egy könyvet olvas Zoli arról, hogy olvas Maria egy könyvet arról, hogy Tibor olvas egy könyvet arról,... (Anna is reading a book about that Zoli is reading a book about that Maria is reading a book about that Tibor is reading a book about that ..)

In this example, the order of some words within the repeating subsentence can be arbitrary, as the semantic role is encoded in Hungarian language with case-making (inflection) of the words. The word “könyvet” is the accusative form of the stem “könyv” (book). Thus all of the following sentences can be used:

- hogy Anna könyvet olvas
- hogy könyvet olvas Anna
- hogy Anna olvas könyvet

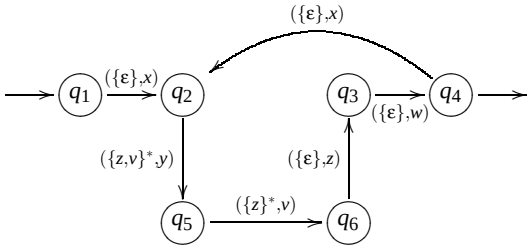

Figure 8: The graphical representation of the DFAwtl for $(x(yzv)^!w)^*$.

- hogy olvas Anna könyvet

The pattern of the corresponding subsentence can be given with

$$(x(yzv)^!w)^*$$

where the $s^!$ symbol denotes here the permutation of the elements of s. In general, the form of the sublanguage is

$$(S_1S_2^!S_3)^*$$

The pattern $(x(yzv)^!w)^*$ can be validated with the NFAwtl grammar given in Figure 8.

Theoretically, the language of pattern $(S_1S_2^!S_3)^*$ with finite S_2 could be accepted also by DFA or by regular grammar. This DFA/grammar should contain the explicit description of every possible permutation. Thus the size of the grammar is $O(|S_2|!)$ times larger than the size of the corresponding *NFAwtl*. Thus the main benefit of *NFAwtl* is the compact representation form of the grammar.

In natural languages, the semantic role can be represented at the syntax level by different ways. The two most usual encoding methods are the inflection and the relative position of the words. In Hungarian language, where free orders can be accepted, the different permutations usually convey different marginally semantic contents like emphases or the opinion of the sender. For example, taking the sentence

Maria sokat olvas (Mary reads a lot)

the following permutations can be constructed:

- Maria sokat olvas (correct, natural)
- Maria olvas sokat (rare use, special situation)
- olvas Maria sokat (special situation)
- olvas sokat Maria (special situation)
- sokat Maria olvas (very special situation, sounds strange)
- sokat olvas Maria (correct, natural)

The complexity of natural languages can be well demonstrated with the phenomena that acceptance of a permutation depends on the semantic of the situation. In the next example a similar sentence is used:

Maria olvas egy könyvet (Mary reads a book)

The related permutations are

- Maria olvas egy könyvet (correct)
- Maria egy könyvet olvas (correct)
- olvas egy könyvet Maria (correct)
- olvas Maria egy könyvet (correct)
- egy könyvet Maria olvas (sounds strange)
- egy könyvet olvas Maria (correct).

There are some cases also in Hungarian where the order of the words has a key role in correct interpretation of the sentence. Let us take the following example:

Péter segítette Jóskát (acc) Tomit (acc) kifesteni (Peter helped John to paint Tom)

The sentence

Péter segítette Tomit (acc) Jóskát (acc) kifesteni (Peter helped Tom to paint John)

means on the other hand a very different situation.

5 CONCLUSIONS

A large group of natural languages has no single dominant word order, several permutations of the symbols are grammatical. The traditional finite automata represent every possible orders explicitly resulting in a huge/complex structure. The presented finite automata with translucent letters can be used for a more compact modeling of free word order in natural languages. The NFAwtl automaton provides a more expressive and precise description of the grammar than

the base (regular) finite automaton. The next step of the investigation is to cover also the non-regular and non-context-free elements of natural languages, since DFAwtl's and NFAwtl's are good candidates to handle some of these features...

Other main result is that we provide a linear time algorithm for the word problem. The considered algorithm is non-deterministic for NFAwtl and deterministic for DFAwtl. Thus, it remains open to give an effective deterministic algorithm for NFAwtl languages.

It is also an interesting question whether all languages accepted by DFAwtl, can be accepted by such a way that all the input letters are erased during the computation.

ACKNOWLEDGEMENTS

The publication was supported by the TÁMOP-4.2.2/C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

REFERENCES

Chomsky, N. (1957). *Syntatic Structures*. The Hague: Mouton Co.

Dryer, M. S. (1997). On the six-way word order typology. *Studies in Language*, 21:69–103.

Dryera, Matthew, S., Haspelmath, and (eds.), M. (2012). *The world atlas of langauges structures*. Max Planck Digital Library, Munich.

Gazdar, G. (1982). Natural languages and context-free languages. *Linguistics and Philosophy*, 4:469–473.

Hockett, C. (1955). *A manual of phonology*. Waverly Press, Baltimore.

Hopcroft, J. E. and Ullman, J. D. (1969). *Formal languages and their relation to automata*. Addison-Wesley Longman Publishing Co., Boston.

Joshi, A. K. (2010). *Mildly Context-Sensitive Grammars*. http://www.kornai.com/MatLing/mcsfin.pdf.

Kornai, A. (1985). Natural languages and the chomsky hierarchy. *Proc. of EACL'85*, pages 1–7.

Matthews, R. J. (1979). Are the grammatical sentences of a language a recursive set. *Synthese*, 40:209–224.

Mery, B., Amblard, M., Durand, I., and Retoré, C. (2006). A case study of the convergence of mildly context-sensitive formalisms for natural language syntax: from minimalist grammars to multiple context-free grammars. *INRA Rapport de recherche*, page nr 6042.

Nagy, B. and Otto, F. (2011). Finite-state acceptors with translucent letters. *BILC 2011 – 1st International Workshop on AI Methods for Interdisciplinary Research in Language and Biology, ICAART 2011 – 3rd International Conference on Agents and Artificial Intelligence*, pages 3–13.

Nagy, B. and Otto, F. (2012a). CD-systems of stateless deterministic R-automata with window size one. *Journal of Computer and System Sciences*, 78:780–806.

Nagy, B. and Otto, F. (2012b). On globally deterministic CD-systems of stateless R-automata with window size one. *International Journal of Computer Mathematics*, pages online first, doi: 10.1080/00207160.2012.688820.

Reich, P. A. (1969). The finitness of natural languages. *Language*, 45:831–843.

Sullivan, W. J. (1980). Syntax and linguistic semantics in stratificational theory. *Current approaches to syntax*, pages 301–327.

Wintner, S. (2002). Formal language theory for natural language processing. *Proc. of ACL'02*, pages 71–76.