Journal of Language and Linguistics
Volume 2 Number 2 2003
ISSN 1475 - 8989

A Finite-State Tool for Nonfinite-State Phenomena: the Agreement Automaton

László Drienkó
ELTE University, Budapest, Hungary

View MS Word Version

Abstract

Recursions, embeddings, and various types of dependencies seem to be a most vexatious problem for both linguistic theories and linguistic processing. Traditionally, such problems cannot be sufficiently solved with finite-state automata. (See Chomsky (1957)) A correlation between as few as only two elements is enough to block applicability of finite-state methods. Thus, for instance, the formal language $a_i b_i$ is context-free, not to mention $a_i b_i c_i$, a context-sensitive language. Moving along the Chomsky-hierarchy we encounter languages whose processing requires stronger and stronger computational apparatus. As for natural languages, Chomsky (1957) shows that there are certain types of correlation between linguistic elements which makes it impossible to process English with finite-state machines. However, arguments have been provided, that English is not a context-free language, either. (See Higginbotham 1984). The implication could be that natural (at least English) language processing in general should be done with a machinery stronger than a push-down automaton.

In this paper we propose an apparatus which incorporates a simple way of representing dependencies allowing linguistic processing to be done in time linear in the number of input elements.

We outline a type of finite-state automaton which we call agreement automaton. Input elements for it are simple attribute-value structures. Agreement constraints are defined for final states. Accepting an input sequence is determined by the fulfillment of these constraints.

After the basic definitions we show how non-regular languages can be processed, and that left-, right-, and centre-embedded structures can be handled by agreement transducers.

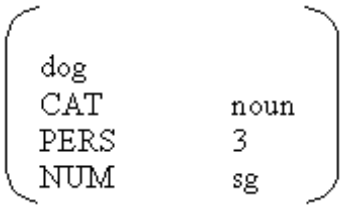
1.The Agreement Automaton

We define an agreement automaton as a 7-tuple:

A (Q, E, b, q0, F, AgrC, T), with

Q:	the finite set of states
E:	the alphabet for input elements
b:	the transition function mapping from Q ´ S to Q
q0:	the initial state (q0 Î Q)
F:	the set of accepting states (F Ì Q)
AgrC(F):	a mapping from states in F to finite sets of agreement constraints/conditions
T(AgrC (F)):	a mapping from sets of agreement constraints to {0,1} (or {failure, success})

Q, b, q0, and F are the usual automaton-components[1]. E is a set of attribute-value structures (AVS) each consisting of attribute-value pairs and a name for reference, e.g.:



Equivalently, E is the set of all names but then we have to specify a mapping from E to the set AVS of all attribute-value structures.

AgrC(F) carries the central concept of our automaton. AgrC(F) specifies for each state q_f in F a set of constraints which have to be satisfied. To put it simply, an input sequence is accepted only if, having read it, the automaton arrives at an accepting state qf and the agreement requirements prescribed by q_f are met.

Each agreement condition consists of three parts. It must specify which elements (states) have to agree, with respect to which attribute, and which agreement strategy must be observed in case of recursion (see below). For instance, constraints:

1 2 4 PERS FIRST-TO-FIRST
1 2 4 NUM FIRST-TO-FIRST
2 5 TENSE FIRST-TO-FIRST

prescribe that elements corresponding to states 1, 2, and 4, respectively, must have the same person and number values, while elements corresponding to states 2 and 5 must have the same TENSE value, and strategy FIRST-TO-FIRST is to be followed.

Formally, it is T(AgrC(F)) that determines the acceptability of an input sequence. If $T(AgrC(q_f)) = 0$ (or $T(AgrC(q_f))$= failure) then the input sequence can not be accepted, whereas $T(AgrC(q_f))=1$ or ($T(AgrC(q_f))$=success) indicates that a grammatical sentence has been recognised.

T(AgrC(F)) can be evaluated by the following algorithm.

T(AgrC(F)) algorithm

0. If accepting state (q Î F) then
1. For all constraints in AgrC(q)
2. Get states to agree, get attribute to check
3. Count the occurrence of sequences, arrange states-to-agree according to type of recursive agreement
4. For all sequences of states to agree
5. For all states to agree

6. If agreement fails return FAILURE (T(AgrC(q)) = 0)
7. Next state
8. Next sequence
9. Next constraint
10. End if
11. Return SUCCESS (T(AgrC(q)) = 1)

The cycle corresponding to lines 4 and 8 is necessary because of possible recursions. If e.g. processing reaches state q_i and then continues with state q_j then recursively steps back to q_i *k* times continuing with q_j and finally accepting state q_f is reached, and if there is some agreement requirement between q_i and q_j then this requirement must be checked *k* times. In fact, it is this property that conditions recursive dependencies leading us out of the realm of regular languages.[2]

The default strategy for recursive agreement is what we have just outlined. We term it as FIRST-TO-FIRST inasmuch as the first occurrence of q_i must agree with the first occurrence of q_j. Since this strategy reflects the temporal alignment of words uttered or perceived in normal speech we assume that it is the 'easiest' (hence default).[3]

Another possibility is to agree the last occurrence of q_i with the first occurrence of q_j, or, equivalently, the first occurrence of qi with the last occurrence of q_j. We call this LAST-TO-FIRST.

Nevertheless, constraints can be constructed in such a way that they contain the possibility of switching on/off the recursion facility of the agreement process. Switching off recursive agreement may be necessary e.g. for cases when some sequence q_1, q_2, ... q_n is processed recursively and then follows p but only q_i should agree with p with respect to some property, where $n > 1$, and $n \geq i \geq 1$. A linguistically motivated case seems to be exemplified by recursive possessive phrases like e.g.: Joe's brothers' friends' house is big.

Here the verb 'be' must agree in person and number with only the last noun, no matter how many other nouns precede it. Since in this case i=n, this type of agreement is referred to as LAST-TO-ONE. A similar type with i=1 would be termed as FIRST-TO-ONE.

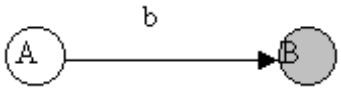
Note that the decision time for T(AgrC(F)) remains linear in n, the length of the input sentence. Indeed, the time needed by the algorithm is basically determined by the three embedded cycles: 1- 9, 4- 8, and 5-7, i.e. it is proportional to CONSTRAINTS x SEQUENCES x STATESTO.

The number of constraints seems to be independent of n. The maximum number of states to agree is n. This corresponds to the extreme case when all the words in the sentence must share a single property. However, the number of sequences is determined by n, and the states that have to agree. More precisely, $n \geq$ SEQUENCES x STATESTO. To see this, consider the sequence 'aaabbbc', and suppose that a's and b's must be checked. Here, n is 7, STATESTO is 2, and SEQUENCES is 3. Then we have $7 > 3 \times 2 = 6$. Thus,

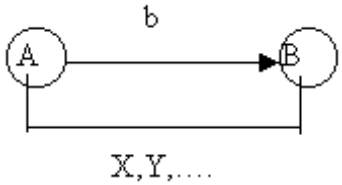
CONSTRAINTS x $n \geq$ CONSTRAINTS x SEQUENCES x STATESTO ~TIME

implies the stated linearity in n.

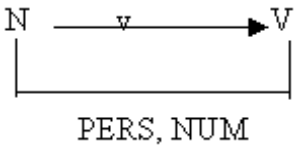
We will use the usual transition diagrams to symbolize transitions between states. A transition from state A to state B on reading 'b' is sketched as:



where 'b' is an attribute value (the CAT value in this paper) of the AVS just read. A shaded circle means an accepting state. Agreement relations between some states A and B with respect to attributes X, Y, ..., are indicated as in the diagram below:

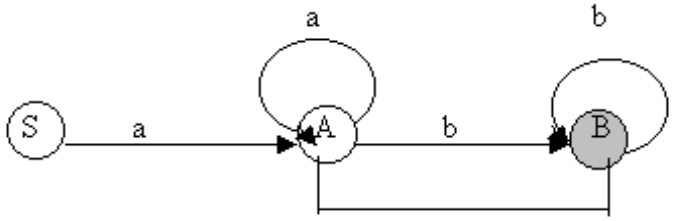


Agreement, in our terminology, means that input elements corresponding to states specified by agreement constraints must have identical values of attributes as prescribed by those constraints. Thus

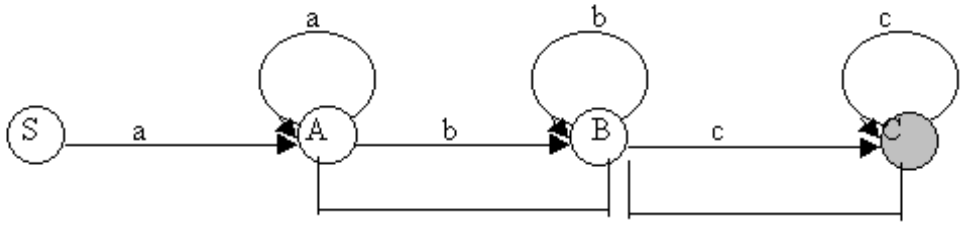


requires that the noun should have the same person and number values as the verb that follows it. However, for the representative examples that we present below there are no linguistically plausible attributes to match. For such cases we may introduce a 'dummy' attribute which can be anything except for attributes already in use. What is important is that the value of this dummy attribute must be the same for all the elements which must agree. This is necessary in order to enable the recursive agreement component of T(AgrC(F)) to detect the occurrence of states properly. For notational convenience in such cases we omit the specification of the dummy attribute below the line symbolizing agreement. As can be expected such cases will relate to recursive phenomena. As particular examples we give automata corresponding to languages a_ib_i, $a_ib_ic_i$.

a^ib^i



$a^ib^ic^i$



Similar methods could be used, e.g., for the type of dependencies implied in languages like $a_ib_jd_ie_j$, $a_ib_jd_je_i$. $a_ib_icd_je_j$ or $a_ib_ica_ib_i$.

The argumentation of Chomsky (1957) against English as a finite state language involves sentence types:

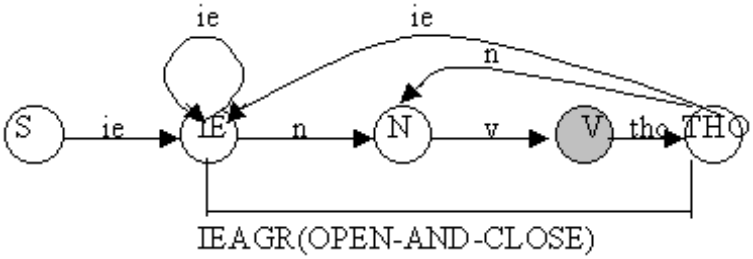
(1.1) If S_1 then S_2
(1.2) Either S_3 or S_4.

Chomsky claims that the interdependency of 'if' and 'then' in (1.1) and that of 'either' and 'or' in (1.2) makes it impossible for a finite state machine to handle recursive combinations of these constructions. Below we show how an agreement automaton can cope with the problem.

First, we introduce word category IE for 'if' and 'either' and category THO for 'then' and 'or'. Then attribute IEAGR is introduced. Next we specify the values of this attribute for the words involved:

if: IEAGR = if_then	then: IEAGR = if_then
either: IEAGR = either_or	or: IEAGR =either_or

Finally, we construct the automaton:

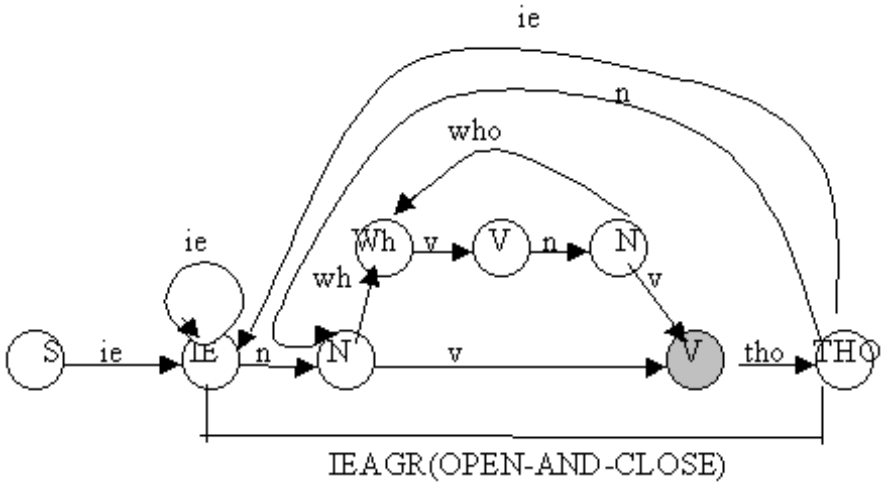


capable of processing sentences like:

(1.3) If Joe sings then either if Peter plays then Ann cooks or if Ann plays then Peter cooks.

To check the proper occurrence of 'if's and 'then's, or 'either's and 'or's we may employ strategy OPEN-AND-CLOSE. As the term suggests, we consider the if-then/either-or problem to be analogous with the problem of establishing the right order of any number of opening and closing brackets. Just like each and every opening bracket must be closed somewhere in a sequence of correctly bracketed elements, each if/either must have its proper then/or counterpart. Note that the appropriate arrangement of the opening and the closing elements as required in line 3 of algorithm T(AgrC(F)) can be done within a single n-cycle, so that the linearity in the number of input elements still holds.[4]

Naturally, relative clause embedding can be incorporated as well. Automaton:



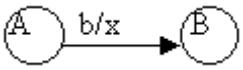
copes with sentences like:

(1.4)1. If Jane who loves Joe who loves Eve works then either Joe who hates Jane works or Eve works

2. Parsing with Agreement Automata: Agreement Transducers

Various attempts have been made to apply finite state methods to parsing natural language sentences. A common approach is to use finite state transducers (See e.g. Kornai (1999) for some insight). Below we show that finite state transducers, supplemented with agreement facilities, can constitute an easy way of handling such most vexatious problems as recursions and embeddings.

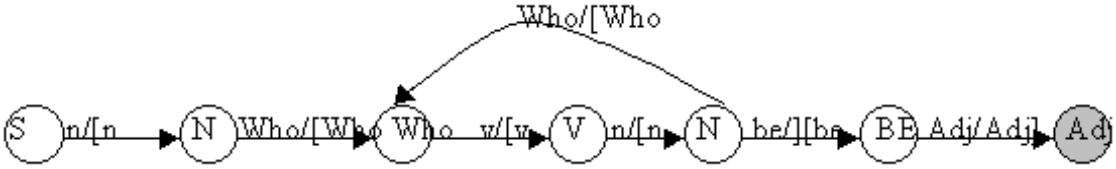
Informally, a transducer is an automaton which can write a symbol onto its output tape. We denote this as



where, 'x' is the symbol to be written on the output tape. Now consider the embedded sentence:

Joe who likes Eve who likes Adam is big.

As a first approximation to parsing such sentences we need a transducer like:



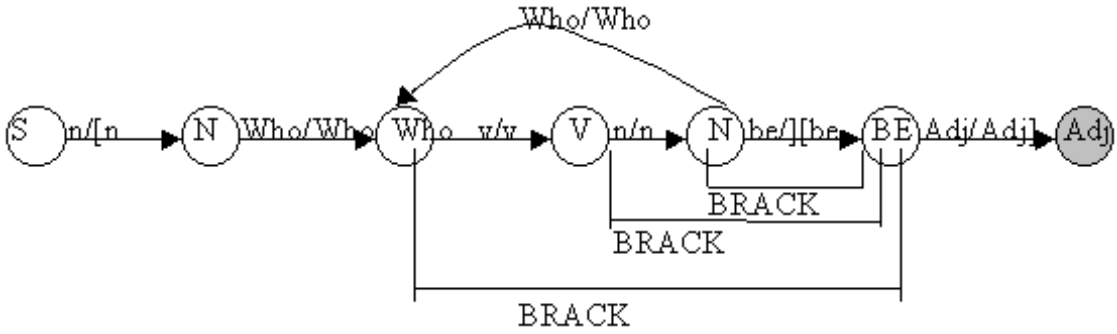
This transducer correctly assigns the opening brackets, however it does not close them properly, yielding the incorrect:

*[n [who [v [n [who [v [n] [be adj]

Note that the decision as to the necessary number of closing brackets can be made no sooner than at state BE. Thus we can argue that there must be some bracketing agreement between state BE and the preceding states where the '['s are to be introduced.

What seems logical is to employ a BRACK attribute whose value is the same for the states in question, and prescribe that the transducer insert the necessary opening and closing brackets any time this agreement constraint is fulfilled. Also, this is an example of 'semi-recursive' ALL-TO-ONE agreement since state BE is processed only once.

Transducer

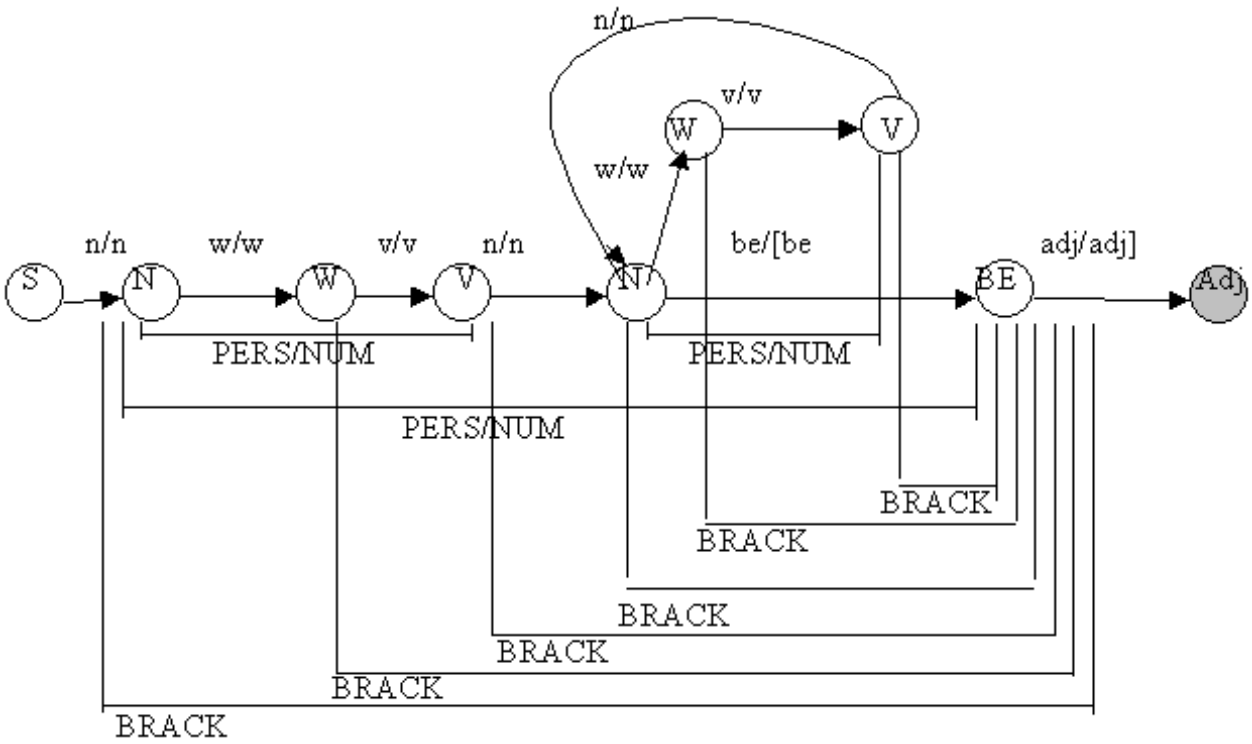


provides

[n [who [v [n [who [v [n]]]]]]] [be adj],

the needed analysis.

To incorporate person and number agreement properties, however, some modifications are necessary. Transducer



can handle sentences of the type:

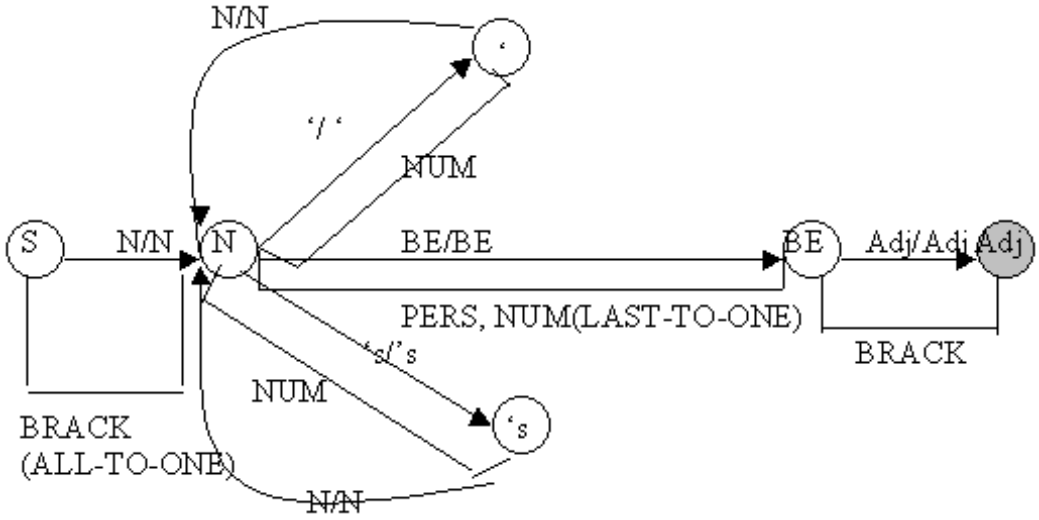
Boys who like Eve who likes girls who like boys are big.

Note that, practically, most bracketing can be done through agreement constraints, so outputting '[be' and 'adj]' on the last two transitions for [be adj] might be replaced with a bracketing agreement between states BE and Adj.

Now we have seen how right-branching structures can be handled. It is easy to see that left-branching constructions can be treated similarly using ALL-TO-ONE agreement. Consider the possessive phrase in sentence

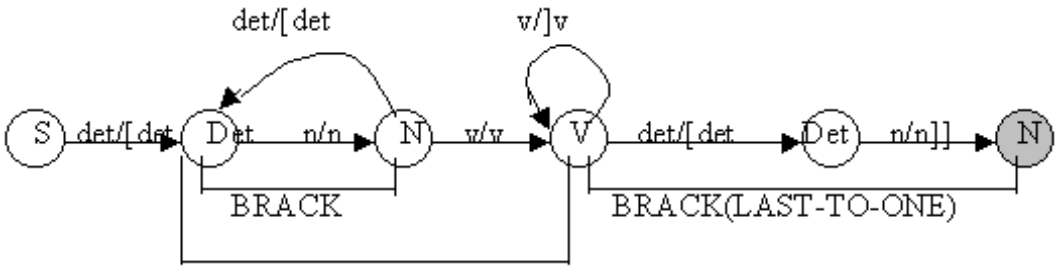
Joe 's friends' friend is big

Transducer



may provide [[[n]'s n]' n] [be adj], the needed structuring.

With center-embedded sentences the above ALL-TO-ONE strategy does not seem to help. Rather, we should construct something like transducer



to manage sentences of type

The rat the cat the dog chased bit ate the cheese

with structure

[[det n] [[det n] [[det n] v] v] [v [det n]]

3. Conclusion

We showed how an agreement automaton can handle dependencies, recursions and embeddings. Our work is in its early stages, so it may need refinement in many ways. Although the basic guidelines are given, we do not exactly know e.g. what attributes are necessary and sufficient to model linguistic phenomena. It is not yet known, either, how full our inventory of recursive agreement strategies is. In a future work we would like to establish the relationship between the statements in this paper and other major issues in syntax. Also, we would like to demonstrate the applicability of our method to other fields.

Notes

1 As we are going to work with word categories rather than with the words themselves, b is actually a mapping Q x CATvalues à Q, that is a mapping from states x word categories to states.

2 In order to check agreement properties we must store the n input elements in an array of size n. This imposes a boundary on possible recursions in practice. In theory, however, this boundary can be arbitrarily long, as it is only dependent on technological factors. On the other hand, we expect

words of languages (of any practical use) to be of finite length. Arguably, it is as pointless to speak about words longer than, say, constant C, as it is no use trying to explore distances shorter than 10^{-35} meters in the (quantum) physical world.

3 Also, this strategy may be reminiscent of languages with cross-serial dependencies where the correlation pattern of linguistic elements is: n_1 n_2 (...nn) v_1 v_2 (...vn). (Identical indices mean correlated elements)

4 We demonstrate this with the following algorithm:

```
r=0, l=0, op=0, times=0 : right(n),left(n)
for i=1 to n
if opening bracket then op=op+1 , r=r+1, right(op)=i
if closing bracket then
l=l+1: if op=0 then failure
left(op)=i
times=times+1: pair(times,1)=right(op), pair(times,2)=left(op)
op=op-1
next n
if r >l or l > r then failure
```

About the Author

Dr Drienko teaches theoretical linguistics at ELTE University, Budapest, Hungary.

Email: dri@axelero.hu

References

Chomsky, N. (1957) *Syntactic Structures*. Hungarian translation: Mondattani szerkezetek Osiris Kiadó, Budapest. 1999.

Higginbotham, J. (1984) English Is Not a Context-Free Language. *Linguistic Inquiry* Vol.15, No.2, pp. 225-234.

Kornai, A (ed.) (1999) *Extended finite state models of language*. CUP, Cambridge.