Finite Automata for Processing Word Order

Wojciech Skut
Universität des Saarlandes, Computerlinguistik
Postfach 151150, 66041 Saarbrücken, Germany
skut@coli.uni-sb.de

Abstract. This paper is concerned with applications of finite state automata (FSA) to word order processing. As the surface order of constituents is a function of many interacting factors, an adequate formalisation seems fairly complicated. I argue that despite this complexity, most ordering constraints observed in free word order languages can be treated by lower-level methods such as finite state automata.

In order to illustrate my claim, I will present a finite state formalisation of some ordering constraints for German. Since the acceptability of an utterance very often depends on the context it occurs in, the approach taken does not only distinguish between grammatical and ungrammatical structures, but also takes into account some discourse information formalised in terms of *information packaging* (ground/focus).

Furthermore, the paper discusses the performance of a grammar of German augmented with an FSA-based word order processing module. Finally, the problem of morphological and syntactic ambiguity is addressed.

1 INTRODUCTION

The current debate on the complexity of natural language has focussed on finding adequate models for certain phenomena (embedding, reduplication, cross-serial dependencies) and investigating their formal properties. This approach makes it possible to distinguish between constructions requiring at least the expressive power of a context-free grammar and those which can be modelled by lower-level methods.

The frequently postulated need for context-free description tools is motivated by different kinds of recursion/embedding: a relative clause can contain another relative clause and so on. Indeed, in order to generate an appropriate functor-argument structure a formalism having the generative capacity of a context-free grammar is required.

However, some phenomena can be handled by less powerful devices. Many aspects of word order fall into this class since the order of constituents depends on their syntactic category and not their internal structure. In certain constructions, the underlying functor-argument structure also matters (cf. center-embedded constructions in standard German or cross-serial dependencies in the Zurich dialect) resulting in a context-free or even -sensitive phenomenon, yet a large number of cases of free word order turn out to be tractable by finite state automata.

Proceedings of the ECAI 96 Workshop
Extended Finite State Models of Language
Edited by A. Kornai.

2 FINITE STATE MODELS OF WORD ORDER

2.1 Regular expressions in grammar rules

Using regular languages for encoding ordering constraints is not a new idea, and has already been incorporated – explicitly or implicitly – into the Functional Unification Grammar (FUG, cf. [3]) and Lexical Functional Grammar (LFG).

For example, the components of English NPs obey word order rules that can be described in the form of the following regular expression:

$$\text{Det Adj}^* \text{ N PP}^* (\text{S}_{\text{rel}})$$

However, generalisations stated in this way are too simplistic as they take into account only categorial information and ignore other relevant factors such as

- pronominality
- focus
- case
- stress

(cf. [6] for a detailed analysis). In a free word order language, it is a combination of these features that can be subject to ordering constraints.

2.2 How free is free word order?

Of course, the expression 'free word/constituent order' does not mean that all permutations of constituents are admissible. Some are ungrammatical, as is the relative order of the two pronouns in the following German sentence:

(1)

*dann	hat	ihn	er	gesehen
then	has	him_{acc}	he_{nom}	seen

'then he saw him'

Others are only allowed in certain contexts (which can themselves be formalised in terms of *focus-ground partitions*[1]; cf. [1]), e.g.

(2)

dann	hat	den Mann	ein Spion	gesehen
then	has	the_{acc} man	a_{nom} spy	seen

'then a spy saw the man'

[1] Also called: *topic-focus* and *theme-rheme*. *Focus* is the part of an utterance that conveys new or communicatively relevant information, while *ground* links this information to the preceding discourse.

is acceptable only if 'den Mann' is ground and 'ein Spion' focus. Such cases of marked word order can be encoded as a binary relation holding of pairs of word order relevant categories, cf. [5].

Some ordering regularities are of a general nature (e.g. SOURCE < GOAL), but others seem to be bound to certain lexical entries, as does the relative order of subjects and indirect objects: while for most German verbs SUBJ < IOBJ (indirect object) is unmarked, cf. (3), the complements of 'unterlaufen' ('happen-to') occur in reverse order, cf. (4).

(3)	dann	hat	der Mann	einem Spion	geholfen
	then	has	the_{nom} man	a_{dat} spy	helped
	'then the man helped a spy'				
(4)	dann	ist	dem Spion	ein Fehler	unterlaufen
	then	is	the_{dat} spy	a_{nom} mistake	happened
	'then the spy made a mistake'				

In general, we can determine whether the relative order of two arguments of a verb is the marked one only if we know the verb itself, or at least the class of verbs it belongs to.

2.3 A model of German word order

Our approach is to define one automaton for each pair of categories whose relative order should be 'supervised' (i.e., a SUBJ-OBJ FSA for the subject and the direct object, as well as an OBJ-IOBJ one for the direct and indirect object, etc.). In every step, the automaton consumes one constituent. Its alphabet thus consists of a finite number of word order relevant categories, i.e., conjunctions of word order relevant features such as grammatical function, pronominality, etc.

Ordering information and the word order relevant category of the constituents under consideration can be expressed by the states (i.e. one state would correspond to the order 'pronominal subject < non-pronominal object', another one to 'non-pronominal object < pronominal subject' etc.). Having encountered the verb, the FSA jumps to a state denoting a marked or an unmarked order, i.e. a particular focus-ground partition. A fragment of the SUBJ-IOBJ automaton is given in figure 1; VSIO stands for verbs whose complements in the unmarked case occur in the SUBJ < IOBJ order, and VIOS for verbs exhibiting the opposite order of arguments.

The automaton supervises the relative order of the subject and indirect object in the so-called Mittelfeld, i.e. the main part of the clause delimited by the finite verb in the V2 position and the clause-final verb cluster (or alternatively: the complementiser and the verb cluster in subordinate clauses). Constraints concerning fronted and extraposed constituents are not modelled by the automaton. For space reasons I have omitted transitions corresponding to clauses starting with a VIOS verb.

For categories other than those on the arcs, the automaton remains in its current state. For instance, adjuncts are consumed without the state being changed.

The states of the FSA encode the following information:

- the presence or absence of the subject and indirect object in the portion of the input already processed by the automaton
- their relative order
- sortal information about the verb

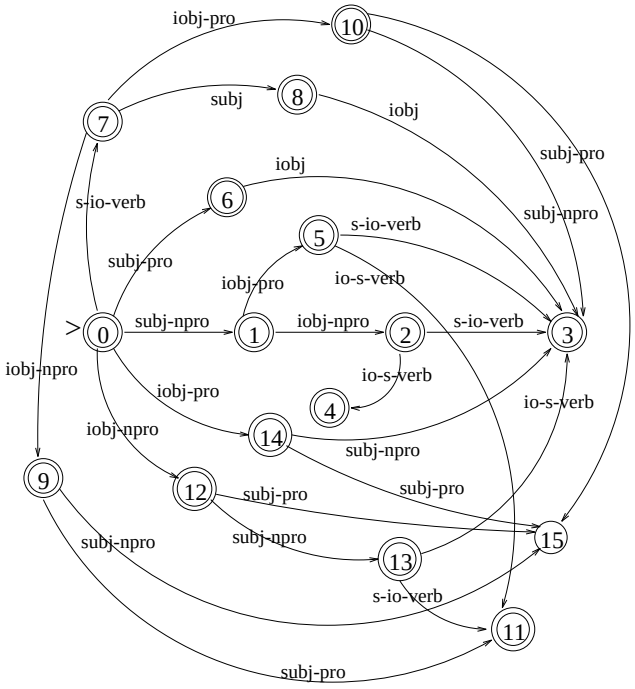

Figure 1. The SUBJ-IOBJ automaton

- information structure

So 10 stands for a situation such that a pronominal object follows a (clause-initial) VSIO verb (e.g. *helfen* - 'help', *gefallen* - 'please').

The initial state is 0. All states except 15 are accepting.[2] 15 stands for ungrammatical word order (e.g., a pronominal subject following an indirect object, cf. the transition 12 → 15). As soon as such a violation of a word order rule is recognised, the automaton jumps to state 15 and remains there. Notice that this strategy is capable of reducing the search space and may increase the efficiency of parsing (see section 2.4).

The states 3, 4 and 11 stand for the information about focus that can be inferred from the relative order of the subject and indirect object. 3 corresponds to the unmarked order and does not say anything about the underlying information structure, while 4 entails that the object is focus and the subject ground, as in example (2) above. 11 stands for focus solely on the subject, as in:

(5)	dann	hat	dem $\text{Mann}_{\text{F-}}$	ein $\text{Spion}_{\text{F+}}$	geholfen
	then	has	the_{dat} man	a_{nom} spy	helped
	'than a spy helped the man'				

The other accepting states do not convey any information about focus since they correspond either to an intermediate stage of processing or to a Mittelfeld containing only one of the two constituents.

[2] Notice that 0, 8, 10, 12 and 14 correspond to a locally incomplete argument structure, but the missing argument of the verb can be fronted or extraposed, hence it seems reasonable to assume that the completeness of the valence frames is taken care of by another automaton which processes constituents occurring both inside and outside the Mittelfeld.

2.4 Integration with the parser

As mentioned above, the word order automata perform two functions, namely

- infer information about focus from the relative order of constituents
- detect and reject ('filter out') strings that violate the corresponding word order constraints

In order to fulfill these tasks, the grammar and the finite state word order module may work in a sequential fashion, the parser proposing one or more analyses of a string, and the automata licensing those not violating German word order rules. In other words, the FSA would be a postprocessor.

However, the 'filtering' functionality of the word order module suggests that there may be some improvement in efficiency in case the automaton is run incrementally and in parallel with the parser. This enables early detection of ungrammatical substructures – both for ill-formed input and wrong local analyses of words or phrases, e.g.

(6) dann gab das Kind es dem Mann
then gave the$_{nom/acc}$ child it$_{nom/acc}$ the$_{dat}$ man
'then the child gave it to the man'

In (6), both *das Kind* and *es* can be either accusative (i.e., the direct object) or nominative (subject). However, if both appear in the Mittelfeld, there is only one reading, namely $nom < acc$. If the parser goes for the other analysis, the SUBJ-OBJ word order automaton will recognise it (and block the whole parse) having processed the pronoun *es*.

The choice of the parallel or the sequential mode may also depend on the character of the language for which the finite state word order module is designed. Languages of the agglutinative or inflectional type exhibiting a rich system of morphological marking (Turkish, Finno-Ugric, Slavic) have a much more restricted ambiguity potential than German, and it cannot be asserted that the parallel model would indeed enhance the efficiency of parsing. However, this issue is still open to investigation.

On the other hand, languages typologically similar to German combine fairly free word order with a highly ambiguous inflectional marking system, so that the parser is confronted with locally consistent partial analyses which however turn out to be ill-formed at the phrasal/sentential level. The scale of this phenomenon allows us to assume that early detection of such inconsistencies may lead to a significant efficiency improvement. This assumption is corroborated by the experimental results described in the next section.

3 IMPLEMENTATION

Until very recently, large-scale grammars of German either restricted the word order in the Mittelfeld to one admissible ordering or permitted all permutations of the Mittelfeld constituents. It is clear that the former approach is too restrictive, and the latter results in overgeneration, thus decreasing parsing efficiency.

The possible improvement can be illustrated by the following experiment. Two versions of a grammar for a fragment of German were compared, each implemented in the ALEP-0 unification formalism (cf. [4]). Both of these grammars use the same rules and lexicon; the only difference is that Grammar 1 additionally exploits a finite-state linear precedence (LP) module.

As for the integration of the LP module with the grammar, the following is of importance:

- the grammar exploits uniform binary-branching structures
- each local binary tree must be licensed by a FS transition characterised by two states (one associated with the head daughter and one with the mother node) and the category of the non-head daughter, which is actually the portion of input consumed in the transition
- the rules have been compiled to several instances, each corresponding to a transition, e.g. the one in figure 2 below is

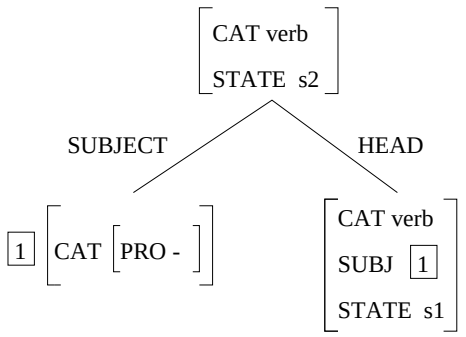

Figure 2. An instance of the subject-head rule

an instance of the head-subject rule unified with a transition from a state s_1 to s_2 over a non-pronominal subject NP (as the system employs a head-driven parser, the transitions do not correspond exactly to those shown in figure 1). The states s_1 and s_2 belong to the product of the state sets of several LP automata, each checking for violations of one ordering rule (SUBJ-IOBJ, OBJ-IOBJ, etc.).

A comparison of the runtime behaviour of the grammars for a few sample sentences is given in the table below:

Table 1. Runtime behaviour of Grammar 1 and Grammar 2

Input	Grammar 1	Grammar 2
gibt er es ihr	0.73 sec.	1.23 sec.
daß es der Mann sieht	4.10 sec.	6.91 sec.
gibt ihm der Mann das Buch	6.75 sec.	38.43 sec.
*gibt es ihr er	0.73 sec.	1.21 sec.

Of course, these performance measurements can serve as an illustration rather than as reliable results since the approach taken was tested only with a small grammar and lexicon. Moreover, the way the automata interact with the grammar is not the most efficient interaction model.

However, the results are encouraging and show that the use of finite-state models of word order rules is a benefit for the efficiency of processing and provides a framework allowing us to capture the complex relationship between word order and other phenomena, e.g. focus.

4 WORD ORDER AND AMBIGUITY

4.1 Main sources of ambiguities

The automaton sketched in figure 1 is deterministic provided the constituents to be parsed are correctly recognised as subjects, objects etc. As mentioned in the previous section, this is often not the case, and the parser has to try different analyses due to morphological and syntactic ambiguities. In other words, we have a deterministic FSA operating over an ambiguous alphabet. The main sources of this ambiguity are the following linguistic phenomena:

- ambiguous morphology, particularly case syncretism:

(7) daß Peter das Kind kennt
then $\text{Peter}_{\text{nom/dat/acc}}$ $\text{the}_{\text{nom/acc}}$ child knows
'that Peter knows the child'

- ambiguity resulting from the fact that the grammatical function of a word is not always uniquely determined by its syntactic case, cf. (*Temp* stands for 'temporal adjunct'):

(8) darum hat er den ganzen Tag beschrieben
that's why has he $[\text{the whole day}]_{\text{Obj}}$ described
'that is why he described the whole day'

(9) darum hat er den ganzen Tag getanzt
that's why has he $[\text{the whole day}]_{\text{Temp}}$ danced
'that is why he danced all day'

- attachment ambiguities

It seems that at least some of these ambiguities can be resolved by taking into account word order information (cf. example (6) in section 2.4). The implementation described in the previous section follows a trial-and-error approach: whenever an ambiguous structure occurs, the parser creates a choice point and tries out all possibilities non-deterministically.

This strategy stems from the inherent ambiguity of natural language. The parser has to reflect it, and hence some non-determinism will always be present in syntactic processing. Nevertheless, different methods of coping with this problem have been developed, e.g. the Tomita algorithm, preference-driven parsing etc. I will now demonstrate a few similar tools that can be used for handling morphological and syntactic ambiguity in the finite-state word order module. However, as they have not been tested yet, the reader should keep in mind that this is just a first approach to the problem.

4.2 Coping with case syncretism

An alternative to the trial-and-error approach is to explore in parallel all transitions corresponding to the readings of an ambiguous constituent. Of course, this will only work in a system exploiting a parallel parsing strategy - also for the deep parser. The word order automata would then provide additional control information for parsing, e.g. by pruning away some branches of the resulting packed parse forest.

A different – but related – method is to use an automaton that would work deterministically even for ambiguous input. Notice that despite the considerable ambiguity potential coming from the fairly weak German case marking, most examples of case syncretism are representatives of a very limited number of classes, among others the following:

- *nom* ∨ *acc*, e.g. *das Fenster* ('the window')
- *nom* ∨ *dat* ∨ *acc*, e.g. *Peter*
- *gen* ∨ *dat*, e.g. *der Frau* 'the woman'
- *nom* ∨ *gen* ∨ *acc*, e.g. *Männer* 'men' (plural forms)

Therefore designing a deterministic automaton does not necessarily imply a much larger number of transitions and states. For instance, it is possible to add to the automaton shown in figure 1 only a few transitions and states, each corresponding to one of the ambiguous categories listed above. This strategy results in a rather modest growth of the device, as shown in figure 3.

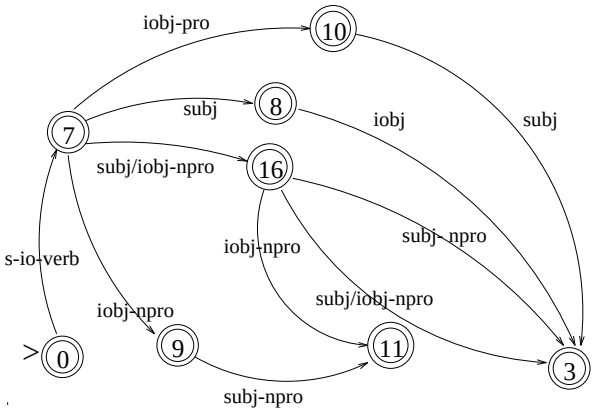

Figure 3. A fragment of the extended SUBJ-IOBJ automaton

The new state 16 corresponds to a substring beginning with a S-IO-verb followed by an ambiguous phrase that can be both subject and indirect object, e.g. a proper noun like *Peter*.[3] If then an unambiguous indirect object is found as in:

(10) dann half Peter einem Spion
then helped $\text{Peter}_{\text{nom/dat/acc}}$ a_{dat} spy
'then Peter helped a spy'

the automaton goes to state 3 as SUBJ (=*Peter*) < IOBJ (=*dem Spion*) is the unmarked order of the arguments of *half*.

Similarly, if an unambiguous subject is found, *Peter* turns out to be the indirect object and the new state of the automaton is 11, which stands for focus on the subject, cf.

(11) dann half Peter ein Spion
then helped $\text{Peter}_{\text{nom/dat/acc}}$ a_{nom} spy
'then a spy helped Peter'

Finally, if another ambiguous constituent is found, e.g.:

(12) dann half Peter Gerda
then helped $\text{Peter}_{\text{nom/dat/acc}}$ $\text{Gerda}_{\text{nom/dat/acc}}$
'then Peter helped Gerda'

we obtain another case of unmarked word order as the first constituent in such a structure is always the subject. Thus the automaton goes to state 3.

[3] I am oversimplifying here – and in the following examples – as *Peter* can also be a direct object.

Of course, the above example is simplified, and hand-coding such automata does not seem realistic. Instead one might consider compiling the deterministic power set automaton and removing those transitions which do not correspond to any of the ambiguity classes given at the beginning of this section.

However, it is difficult to estimate the possible efficiency gain that can be achieved using this method as long as it has not been tested with a large-scale grammar and corpora containing different types of sentences. Therefore, other strategies of coping with morphological and syntactic ambiguity should also be taken into consideration.

4.3 Preference-driven parsing

Another way of dealing with ambiguities is to proceed non-deterministically, but explore the arising alternatives in an order determined by some additional information – e.g. parsing preferences expressed by weights associated to the transitions of the automaton. Thus a construct similar to hidden Markov models would be created. The values of the weights could be retrieved from corpora annotated with what I have called *word order relevant information* in this paper.

If we choose this strategy, we could also consider replacing the traditional tree-based phrase structure model of syntax by a dependency-oriented one, cf. [2]. Such an approach may prove efficient at handling different kinds of long-distance dependencies and discontinuous constituency, as the preference values provide control data that can constrain the search space of the parser.[4] Therefore we would avoid the exponential search complexity widely considered the most serious problem for dependency parsers.

5 RIGID WORD ORDER

In section 2, I hinted that finite automata can be used to encode ordering constraints in phrases exhibiting rather rigid word order, e.g. English or German NPs. The encoding is fairly easy compared with the German Mittelfeld, yet it is still worth noting that by adopting a finer-grained notion of syntactic category, we can can handle not only the relative order of determiners and adjectives, adjectives and the noun, etc., but also the partly idiosyncratic ordering regulatities concerning multiple determiners – or, more generally, prenominal modifiers, cf.

(13)	all	die	vielen	Spieler
	all	the	many	players

While (13) is grammatical, the combinations **die all vielen Spieler*, **die vielen all Spieler*, **die all vielen Spieler* are not. This order is encoded in the FSA in figure 4. The states of the automaton encode the part of the NP consumed so far. We distinguish between different classes of prenominal modifiers

[4] The preference weights play the same role as the adjacency requirement for parsing context-free and ID/LP grammars – for every application of a grammar rule they say where to look for the components of the structure licensed by the rule in question. Predictably, the preference values will sharply decrease for constructions containing too many non-local dependencies, which provides a good guide for parsing. On the other hand, the numerically encoded preference values permit more flexibility than trees in that we are not forced to introduce additional devices (transformations, etc.) for handling non-local dependencies.

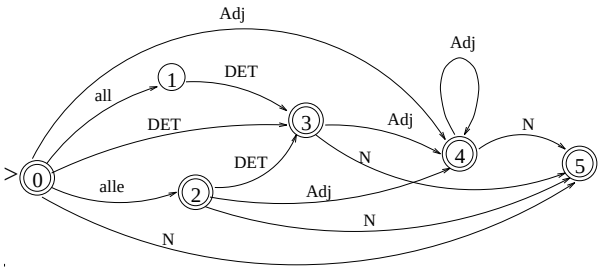

Figure 4. An automaton for processing NPs

(DET for articles and demonstrative pronouns, Adj for adjectives and adjectival quantifiers, etc.). The automaton encodes some further ordering rules and accepts elliptic NP such as *alle*. The ungrammatical NPs given above are rejected.

6 CONCLUSION

The application of finite-state methods to the processing of free word order can be studied from two different perspectives. Firstly, finite automata make a good tool for describing the complex interaction of focus, pronominality and other ordering-relevant factors. Secondly, as word order information often allows us to resolve ambiguities, one may expect an improvement in the efficiency of syntactic processing.

The first results, described in section 3, are promising. However, there are still some open questions concerning the methods of handling ambiguity addressed in this paper, particularly the issue of deterministic finite state processing and preference-driven parsing. Thus the whole section 4 should be treated as an outline of possible solutions and motivation for further discussion rather than completed work.

ACKNOWLEDGEMENTS

The work reported in this paper was supported by the Deutsche Forschungsgemeinschaft in Sonderforschungsbereich 378. I wish to thank Thorsten Brants, Sascha Brawer, Hans Uszkoreit and Christian Weiß for their valuable comments and suggestions.

REFERENCES

[1] E. Engdahl and E. Vallduví, 'Information packaging and grammar architecture: A constraint-based approach', in *Integrating information structure into constraint-based and categorial approaches*, ed., E. Engdahl, 39–79, ILLC, Amsterdam, (1994). DYANA-2 Report R.1.3.B.

[2] Richard Hudson, *Word Grammar*, Basil Blackwell Ltd., 1984.

[3] M. Kay, 'Parsing in functional unification grammar', in *Natural Language Parsing*, eds., D. Dowty, L.Karttunen, and A. Zwicky, Cambridge University Press, Cambidge, UK, (1985).

[4] N. Simpkins, *ALEP-0 Prototype Virtual Machine: User Guide, Version 2.0*, 1992.

[5] W. Skut and C. Weiß, 'Modelling focus projection with finite state automata', in *Proceedings of Traitement Automatique de Langage Naturel*, (1996).

[6] H. Uszkoreit, *Word Order and Constituent Structure in German*, CSLI, 1987.