Finite-State Transducers: Parsing Free and Frozen Sentences

Emmanuel Roche
Mitsubishi Electric Research Laboratories
201 Broadway
Cambridge, MA 02139
roche@merl.com

Abstract. Accurately parsing natural language sentences requires very large scale lexical grammars. Rules of such grammars can roughly be classified as "free" or "frozen" according to the variability of the arguments, e.g. complements. For instance, a rule describing a sentence like *John eats potatoes* would be called free whereas a rule describing a sentence like *John kicks the bucket* would be called frozen. Usually, grammar representations describe frozen rules as a particular case of free ones and assume that they can be treated as exceptions. Finite-state transducers not only give simple and efficient parsing strategies for very large scale grammars but that they also provide a natural and unified way of handling sentences in which free and frozen components are mixed in a complex manner.

In language processing, finite-state models are not a lesser evil that bring simplicity and efficiency at the cost of accuracy. On the contrary they provide a very natural framework to describe complex linguistic phenomena.

1 Introduction

Finite-State methods have recently improved computational efficiency dramatically for a wide variety of natural language processing tasks; ranging from morphological analysis [10, 19, 3] to phonetic and speech processing [15, 11].

However, finite-state modeling is usually thought as a necessary evil in the sense that more powerful formalisms such as context-free grammars are more accurate but of intractable size for reasonable efficiency. A clear illustration of this view comes from the field of speech recognition in which grammar are often given in a context-free form but the size of the data and the finite-state nature of most representations (phoneme or word lattice for instance) make it difficult and inefficient to use general algorithms such as the Earley parsing. Complex grammars are therefore approximated by finite-state models [14], these approximations lead to more efficient and simpler parsing strategies at the cost of a lost of accuracy.

For the problem of parsing natural language sentences, in many respects, finite-state models are not an efficient but somewhat inaccurate tool but rather one of the best formalism at hand to represent accurately complex linguistic phenomena. We take here a complementary approach to [7] which shows that FSA is a very natural framework for the task of designing large coverage and lexicalized grammars. Here we focus on sentence parsing, therefore assuming that a very precise grammar is already available. While our main point is not grammar design, the discussion should shed light on several important problems and solutions involved at the grammar building stage.

One of the main drawback in using context-free grammars for modeling language is the inability or the difficulty of handling various types of deletion. Consider the sentence

(1) *He expected John to buy a new book.*

In the transformational grammar of Z. S. Harris (see [9] for instance), this sentence is analyzed as the verb *expected* (an *operator*) taking three arguments: (1) the subject *he*, (2) the first complement *John* and (3) the sentence *John buys a new book*; the application of the verb operator deletes the subject of the last argument to transform the sentence into the infinitive clause *to buy a new book*. In order to handle this situation with context-free grammars, each sentence has to be described both with its declarative form, say *N buy N*, and with its infinitive clause form, say *to buy N*. This might not seem to be a difficult problem at first but recall that grammars have to be lexicalized (see [5, 2] for instance), that is, each rule should contain an explicit word, and therefore this type of duplication, which is only one duplication among many others, has to be repeated throughout the whole lexicon. While other features of transducer parsing, such as

- factorization, determinization and minimization of transducers can be used to generate more efficient parsers,
- the grammar compilation, input sentences and the parsing use homogeneous representations,
- parsing and building the grammar are the same operation, i.e. a rational transduction,
- transforming a CFG into a transducer is yet another transduction,

have been described elsewhere [17, 18], we show how transducers can be used to parse sentences such as (1) while respecting the transformational analysis.

The next section will give a short background about parsing viewed as a string transformation and about finite-state transducers, the following section, describing one of the ways context-free languages can be parsed with finite-state transducers, give the general framework for transducer parsing. The main section, section 4, shows, through a variety of linguistic examples, that transduction parsing is well adapted to transformational grammars and that, in addition to lead to efficient parsing strategies, it also lead to more accurate sentence analysis.

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

2 Background

Finite-state transducers (FST) are informally (see [1, 16] for formal definitions) finite-state automata for which each transition is labeled by a pair of labels. The left label is called the input label and the right one is called the output label. FSTs are represented by oriented graph such as the ones of Table 2[1]. A FST defines a function from strings into strings in the following way: given an input sequence of symbols, if a path of the FST can be followed while reading the input sequence on the input label, then the concatenation of the output labels of the transitions of this path is an output for the input sequence. For instance, the input sequence [N *John* N] is transformed into (N *John* N) by the third transducer of Table 2.

Parsing will be considered here as a string transformation process. Namely, if Σ_g represents the set of symbols, such as (N, N) or (S, used to mark the syntactic analysis; if Σ_w is the list of words in the language, then a parser is a mapping

$$\text{parser} : \Sigma_w^* \longrightarrow 2^{(\Sigma_g^* \cdot \Sigma_w^* \cdot \Sigma_g^*)^*}$$

such that, for instance, a sequence of words like[2]

(2) *John left this morning.*

is transformed into a set of of outputs (here the set contains only one element) representing the analysis of this sentence:

(3) (S (N *John* N) (V *left* V) (N *this morning* N) S)

3 A Top-Down Parser for Context-Free Grammars

One way of parsing context-free grammars with finite-state transducers consists of modeling a top-down analysis. Consider the sentence

(4) *John thinks that Peter kept the book*

and suppose the we have the syntactic data of Table 1.

N *thinks that* S	S
N *kept* N	S
John	N
Peter	N
the book	N

Table 1. syntactic dictionary for *John thinks that Peter kept the book*

This data can be seen as a sample of a syntactic dictionary in which the keys are the structures and the information is the type of the structure (sentence structure, noun structure).

[1] On this graph, the symbol *?* stands for any symbol in the alphabet considered and the symbol <E> stands for the empty word ϵ.

[2] In reality, parsing apply on a sequence of symbols representing the result of the morphological analysis. Such representation contains both the words of the input sentences and morphological features such as number, gender, etc. Here, to simplify the exposition, we make the hypotheses that the morphological information are available when necessary. The details on how morphological analysis and parsing with transducers are combined is described in [17].

The first step of building a parser consists of transforming each entry of this syntactic dictionary into a finite-state transducer. The finite-state transducer related to an entry will be the machine responsible for analyzing a sentence with this structure. Here, this will lead to the transducers of Table 2 which can be seen as a dictionary of transducers.

Each transducer represents a transduction from Σ^* to Σ^* where $\Sigma = \Sigma_w \cup \Sigma_g$. For instance, the transducer associated to the structure N *thinks that* S (the first transducer of Table 2), that we denote T_{thinks_that}, will map (5) to (6).

(5) [S *John thinks that Peter kept the book* S]
(6) (S [N John N] thinks that [S Peter kept the book S] S)

Formally, $T_{thinks\_that}(5) = (6)$.
Given the dictionary we define the grammar

$$T_{dic} = \bigcup_{T_i \in dic} T_i$$

as being the union of all the transducers defined in the dictionary. For instance, if dic_1 is the dictionary defined in Table 2, then T_{dic_1} is the transducer represented of Figure 1.

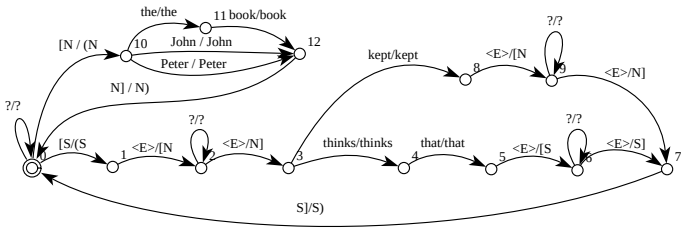


Figure 1. Transducer T_{dic_1}

This transducer being given, parsing simply consists of applying the transducer on the input and checking whether the output is different from the input; if it is, then the transducer is applied again. The process is repeated until the input is not modified. Formally,

$$\text{parser} = T_{dic}^{\infty}$$

On the example of the input (5), the parsing is illustrated below:

[S John thinks that Peter kept the book S]
$\downarrow (T_{dic_1})$
(S [N John N] thinks that [S Peter kept the book S] S)
(S [N John thinks that Peter N] kept [N the book N] S)
$\downarrow (T_{dic_1})$
(S (N John N) thinks that (S [N Peter N] kept [N the book N] S) S)
$\downarrow (T_{dic_1})$
(S (N John N) thinks that (S (N Peter N) kept (N the book N) S) S)

The input sequence is given on top and the first application of T_{dic_1} results in two outputs. We reapply the transducer on each one, the first one leads to the next sequence while the second one has no output. Finally, the latest sequence is:

(7) (S (N John N) thinks that (S (N Peter N) kept (N the book N) S) S)

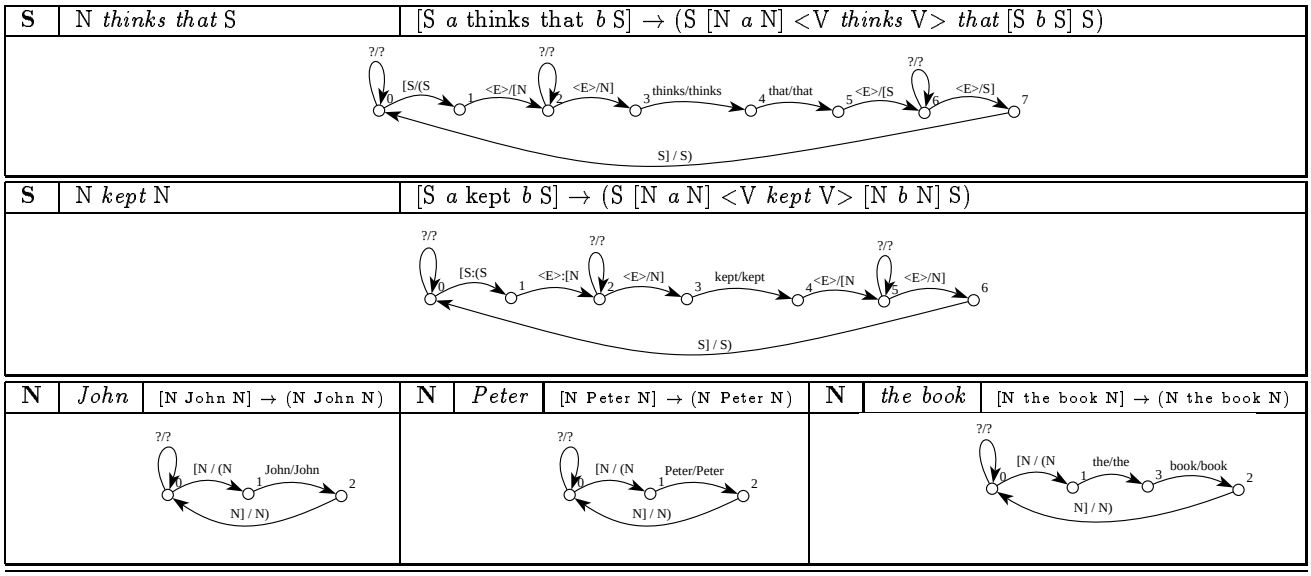


Table 2. Transducers associated to each structure

which is also the analysis of the input.

The parsing is described here as the application of a transducer on a string however, in practice, the input strings are represented by FSAs and the transducers therefore apply on these FSAs directly.

4 Transformation Grammars

[6] demonstrates that three categories of sentences have to be considered to build complete grammars. The first category consists of free sentences like *John eats potatoes*, the second one consists of sentences like *John makes concessions*, with complements with a smaller degree of variability. They are called *support verb constructions*. Finally, the third category consists of *frozen sentences*, or idiomatic expressions, such as *John kicks the bucket*, in which one or several arguments, e.g. complements, are fixed. This section presents a few typical problems encountered while handling sentences of one of the last two categories.

4.1 Support Verb Constructions

The following sentences are example of *ssupport* (or *light*) verb constructions.

(8) *John makes concessions to his friend.*
(9) *John makes a right turn*
(10) * *John makes a right turn to this friend.*
(11) *John's concessions to his friend were unexpected.*

If *make* is analyzed as a verb such as *read* then sentences (8) and (9) should be analyzed with the structures

(12) N_0 *makes* N_1 to N_2
(13) N_0 *makes* N_1

This analysis takes the verb as the head of the sentence but fails to explain why sentence (10) is forbidden.

Furthermore, (8) is clearly present in sentence (11). In fact, the noun phrase construction *N's N to N* is not general and

(14) * *John's turn to his friend.*

is clearly forbidden too. These two observations, among others (see [4]), lead to analyze sentences such as (8) with the noun (called *predicative noun*) as the real head and the verb, called *support verb* (or *light verb*), as support for the tense. (8) will therefore be described by the structure

(15) N_0 (V_{sup} make V_{sup}) (N_{pred} concessions N_{pred}) to N_1

The diversity of the following examples, with the support verb *take*, further illustrates that the predicative noun, and not the verb, governs the number and the nature of the arguments.

(16) *John takes a decision.*
(17) *John takes a look at this.*
(18) *John takes advantage of his position.*
(19) *John takes credible steps toward solving this problem.*
(20) *The party is not likely to take a backseat.*

To be able to parse such sentences each of the following transducers[3], corresponding respectively to (8), (16), (17) and (18) should be added to the global syntactic dictionary

$[S\ a\ make\ concessions\ to\ b\ S]$ →
(S [N a N] < V_{sup} make V_{sup} > < N_{pred} concessions N_{pred} > to [N b N] S)

$[S\ a\ take\ a\ decision\ S]$ →
(S [N a N] <V_{sup} take V_{sup}> <N_{pred} a decision N_{pred}> S)

$[S\ a\ take\ a\ look\ at\ b\ S]$ →
(S [N a N] <V_{sup} take V_{sup}> < N_{pred} a look N_{pred}> at [N b N] S)

$[S\ a\ take\ advantage\ of\ b\ S]$ →
(S [N a N] < V_{sup} take V_{sup} > < N_{pred} advantage N_{pred} > of [N b N] S)

[3] Here, the transductions are defined functionally but in practice they are defined by their graph representation.

4.2 Support Verb and Sentential Clause

In sentence (19), the construction *John takes credible steps toward* is followed by a sentence clause whose verb's subject is also the subject of the main clause, that is, *John*. The analysis works as follows: the input sequence for the parsing transducer is

(21) [*S* John takes credible steps toward V_{ing} solving the problem *S*]

in which V_{ing} is a marker generated by the morphological analysis.[4]

We build the grammar such that the first application of the transduction to (21) leads to the following sequence:

(22) (*S* [*N* John *N*] <V_{sup} takes V_{sup}> <N_{pred} credible steps N_{pred}> toward $OP_{N_0 \rightarrow NS}$ [*S N* $V_{(ing)}$ solving the problem *S*] *S*)

This performs simultaneously the following eight actions:

- the subject is enclosed into [*N* and *N*] brackets,
- the sequence *take credible steps toward* is recognized,
- *take* is marked as a support verb with <V_{sup} and V_{sup}>,
- *credible steps* is marked as a predicative noun with <N_{pred} and N_{pred}>,
- *solving* is recognized as a verb in *ing* form,
- *solving this problem* is marked as a sentence to be recognized while a subject marker *N* is added.
- The morphological marker V_{ing} is transformed into the marker $V_{(ing)}$ to signify that although the verb was originally in the *ing* form, it has to be considered as a simple conjugated verb during the rest of the analysis.
- A marker $OP_{N_0 \rightarrow NS}$ is inserted to link the subject of the sentence with the subject of the *ing* clause, that is the subject of *solving*.

This should be interpreted in the context of transformation grammars in which the sentence is analyzed as the operator *John takes credible steps toward* applying to the sentence *John solves the problem*. The subject of the second sentence is deleted within this operation.

The second step of the analysis consists of analyzing *John* as a nominal and *N* $V_{(ing)}$ *solving the problem* as a sentence. The sentence structure *N solve N* should therefore be compiled into a transduction that takes into account both the possibility for the sentence to appear by itself, e.g. *John solves the problem*, or, as in our example, to appear as an *ing* clause. In order to handle both situations, the grammar should perform the following two mappings:

[*S* a solve b *S*] → (*S* [*N* a *N*] <V solve V> [*N* b *N*] *S*)

[*S N* $V_{(ing)}$ solving b *S*] → (*S N* $V_{(ing)}$ <V solving V> [*N* b *N*] *S*)

The application of the grammar to the sequence of (22), i.e. the second step of the analysis of (21), leads to the following sequence:

(23) (*S* (*N* John *N*) <V_{sup} takes V_{sup}> <N_{pred} credible steps N_{pred}> toward $OP_{N_0 \rightarrow NS}$ (*S N* $V_{(ing)}$ <V solving V> [*N* the problem *N*] *S*) *S*)

and finally to the analysis

(24) (*S* (*N* John *N*) <V_{sup} takes V_{sup}> <N_{pred} credible steps N_{pred}> toward $OP_{N_0 \rightarrow NS}$ (*S N* $V_{(ing)}$ <V solving V> (*N* the problem *N*) *S*) *S*)

4.3 Support Verb Recovery in Noun Clauses

Let us now consider a different sentence:

(25) [*S* John's concessions to his friend were unexpected *S*]

The difficulty is to analyze correctly the nominal *John's concessions to his friend*. Recall that the grammar should not contain a rule that says that the structure $N'sNtoN$ can always form a nominal, this would generate many incorrect analysis. In fact, this type of nominal is made possible here by the underlying support verb construction

(26) *John makes concessions to his friend*

and the analysis should therefore reduce the problem of analyzing the nominal into the analyzing this sentence. The first application of the transducer representing the grammar will transform the original sentence (25) into the following one:

(27) (*S* [*N* John's concessions to his friend *N*] <*V* were *V*> [ADJ unexpected ADJ] *S*)

Linguistically, a sentence, such as (26), of the shape[5]

(28) *N* V_{sup} N_{pred} *W*

can often be transformed into a nominal with the following shape:

(29) *N*'s N_{pred} *W*

in which the support verb disappears. To cover this phenomenon, the following mapping, corresponding to the nominal structure (29) should be added to the grammar

[*N* a 's concessions to b *N*] → (*N* [*S* a V_{sup} ?concessions to b *S*]N)

The transduction representing the structure

(30) *N make concessions to N*

should also perform the two mappings

[*S a make concessions to b S*] →
(S [N a N] < V_{sup} make V_{sup} > < N_{pred} concessions N_{pred} > to [N b N] S)

[*S a* V_{sup} *make concessions to b S*] →
(S [N a N] V_{sup} < V_{sup} make V_{sup} > < N_{pred} concessions N_{pred} > to [N b N] S)

[4] Recall that, in practice, the transduction doesn't apply on the text directly but to the sequence of words and symbols representing the result of the morphological analysis.

[5] *W* represents any number of arguments; the type and number of arguments depend on the predicative noun N_{pred}.

The first one handles sentences such as (26) and the second one validates the support verb construction hypothesis.

With these mappings, the analysis of (25) is performed as indicated on Figure 2. Here again, the underlying support verb sentence is explicitly recovered during parsing and can be extracted from the resulting analysis.

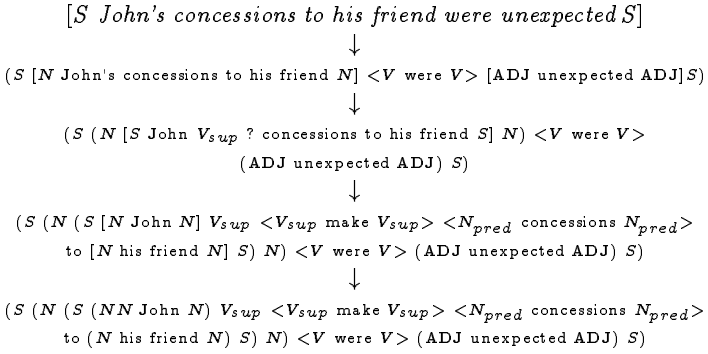


Figure 2. Analysis of the sentence *John's concessions to his friend were unexpected*

4.4 Hidden Support Verb Constructions

Let us now consider the following more complex sentences:

(31) *John asked Peter for an immediate decision*
(32) *John asked Peter for his belongings*
(33) *John asked Peter for a ride*

which share the common surface sentence structure:

(34) N_0 *ask* N_1 *for* N_2

However, describing such sentences only with the surface structure is not sufficient. In fact, such description would not explain why the following sentence

(35) * *They asked Peter for their immediate decision*

is not accepted, contrary to

(36) *They asked Peter for his immediate decision*
(37) *They asked Peter for his belongings*
(38) *They asked Peter for their belongings*

The linguistic explanation is that sentences (31) and (36) are analyzed as transformations of

(39) *They asked Peter to make (a/his) decision immediately*

whereas (37) and (38) are analyzed as transformations of

(40) *They asked Peter to give them (his/their) belongings*

[13] shows that, in French, complex verbs such as *to ask*, should be described in conjunction with a list of support verbs. The support verbs such as *take* and *give* are then deleted by the constraints they impose remain. The situation is almost identical in English. Here for instance, the infinitive of (39) contains the sentence

(41) *Peter makes a decision immediately*

or equivalently,[6]

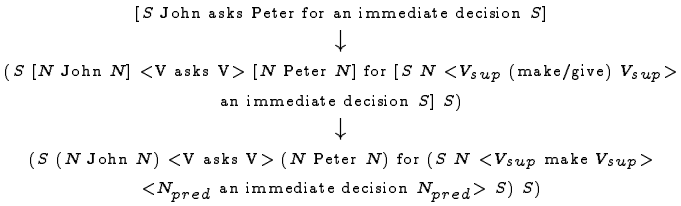
(42) *Peter makes an immediate decision*

The verb *makes* disappears within the transformation. However it is still necessary to know which construction is really used. In fact, the support verb construction allows sentences

(43) *Peter makes his decision*

in which the word *his* has to refer to the subject of the construction. This explains why (35) is incorrect.

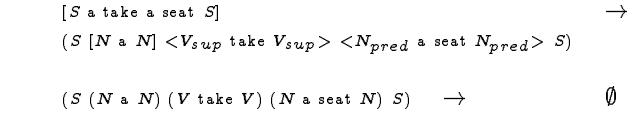
In order to take this properties into account the parsing of (31) works as indicated in the following trace:



Here again, it is possible, within the parsing program, to introduce hypotheses about several support verb constructions. Some of these hypothesis are invalidated at later stage of the analysis and the correct support verb is recovered.

4.5 Frozen Expressions

A recurrent problem when recognizing support verb constructions or frozen expressions is that these sentences also have the surface form of a free expression. For instance, the sentence [*S John take (a/his) seat S*] is going to be parsed in two different ways. First, it will be parsed as a free sentence (by resemblance to [*S John take a seat S*]) and the resulting analysis will be (*S* (N *John* N) <V *take* V> (N *a seat* N). It will also be parsed as a support verb construction and, in that case, the analysis will be (*S* (N *John* N) <V_{sup} *take* V_{sup}> <N_{pred} *a seat* N_{pred}>. Whereas both analysis are possible in principle, the frozen interpretation is usually the correct one. For the analysis to contain the frozen interpretation only, the parser not only has to produce the correct analysis, but it has to delete the incorrect one. In other words the transducer representing the structure *N take a seat* should perform the following two mappings



The first mapping handles the analysis per see whereas the second one deletes the improper analysis resulting from the free structure.

The parsing trace of the sentence is then the following:

[6] This transformation, first described by Z.S. Harris [8], is very common an is even on the criterion to identify a support verb construction (see also [4, 12]).

$$[S\ John\ takes\ a\ seat\ S]$$
$$\downarrow$$
$$(S\ [N\ John\ N]\ <V\ takes\ V>\ [N\ a\ seat\ N]\ S)$$
$$(S\ [N\ John\ N]\ <V_{sup}\ takes\ V_{sup}>\ <N_{pred}\ a\ seat\ N_{pred}>\ S)$$
$$\downarrow$$
$$(S\ (N\ John\ N)\ <V\ takes\ V>\ (N\ a\ seat\ N)\ S)$$
$$(S\ (N\ John\ N)\ <V_{sup}\ takes\ V_{sup}>\ <N_{pred}\ a\ seat\ N_{pred}>\ S)$$
$$\downarrow$$
$$(S\ (N\ John\ N)\ <V_{sup}\ takes\ V_{sup}>\ <N_{pred}\ a\ seat\ N_{pred}>\ S)$$

In the gradation from free to frozen for the verb *take*, we should also consider sentences such as *The elected candidate takes his seat in the House*. This sentence should be interpreted neither as a free construction with an object complement *his seat* and an adverbial *in the House*, nor as the support verb construction *to take (a/his) seat* with the adverbial *in the House* but rather as the frozen expression *N take POSS seat in the House* in which only the subject may vary. For the parser to achieve this analysis, it should perform three mappings:

$[S$ a take his seat in the House $S]$ $\rightarrow$
$(S\ [N$ a $N]$ <F take his seat in the House F> $S)$

$(S\ (N$ a $N)\ (V$ take $V)\ (N$ a seat $N)$ (ADV in the House ADV) $S)$
$\rightarrow$ $\emptyset$

$(S\ (N$ a $N)\ <V_{sup}$ take $V_{sup}>\ <N_{pred}$ a seat $N_{pred}>$ (ADV in the House ADV) $S)$
$\rightarrow$ $\emptyset$

The first mapping performs the analysis per see, the second one deletes the free sentence analysis while the third one deletes the support verb construction analysis.

5 Conclusion

Wide coverage grammars require the linguist and automatic processing programs to handle a myriad of interrelated sentence structures and sentence transformations. A particular formalism can be shown to be better, whether the criterium is naturality, efficiency or anything else, only if applied to the full complexity of language. The problems described here appeared at each step of the design of large scale grammar of French (9,000 free verbs, 6,500 support verb constructions, 5,000 frozen expressions[17]) as well as a medium size English grammar (roughly half of the size of the French grammar). A natural treatment of such sentences was absolutely necessary to the coherence of the whole description. We have shown here that finite-state transducer parsing allows to process transformations, with deletion for instance, for which rewriting mechanisms such as context-free parsing are at best unnatural. In addition, transducer parsing can use explicit negative rules (rules targeted toward deleting wrong interpretations) hence reflecting a very common language behavior.

REFERENCES

[1] Jean Berstel, *Transductions and Context-Free Languages*, Teubner, Stuttgart, 1979.

[2] Alain Guillet Boons, Jean-Paul and Christian Leclere, *La structure des phrases simples en français, Constructions Intransitives*, Librairie Droz, Geneve-Paris, 1976.

[3] David Clemenceau and Emmanuel Roche, 'Enhancing a large scale dictionary with a two-level system', in *EACL-93, proceedings of the conference*, (1993).

[4] Jacqueline Giry-Schneider, *Les prédicats nominaux en français, les phrases simples à verbe support.*, Droz, Genève, Paris, 1978.

[5] Maurice Gross, *Méthodes en syntaxe,régime des constructions complétives*, Hermann, 1975.

[6] Maurice Gross, 'Les limites de la phrase figée', *Langages*, (90), 7–22, (June 1988).

[7] Maurice Gross, 'The construction of local grammars', in *Finite-State Devices for Natural Language Processing*, eds., Emmanuel Roche and Yves Schabes, MIT Press, (1996). Forthcoming.

[8] Zellig Harris, *Notes du cours de syntaxe*, Seuil, Paris, 1976.

[9] Zellig Harris, *Theory of Language and Information*, Oxford University Press, 1991.

[10] Lauri Karttunen, Ronald M. Kaplan, and Annie Zaenen, 'Two-level morphology with composition', in *Proceedings of the 14^{th} International Conference on Computational Linguistics (COLING'92)*, (1992).

[11] Eric Laporte, 'Phonétique et transducteurs', Technical report, Université Paris 7, (1993).

[12] Annie Meunier, *Nominalisations d'adjectifs par verbs supports*, Ph.D. dissertation, Université Paris 7, 1981.

[13] Mehryar Mohri, *Analyse et Représentation par Automates de Structures Syntaxiques Composees*, Ph.D. dissertation, Université Paris 7, January 1993.

[14] Fernando Pereira and Rebecca N. Wright, 'Finite state approximatnion of phrase structure grammars', in *Finite-State Devices for Natural Language Processing*, eds., Emmanuel Roche and Yves Schabes, MIT Press, (1996). Forthcoming.

[15] Fernando C. N. Pereira, Michael Riley, and Richard W. Sproat, 'Weighted rational transductions and their application to human language processing', in *ARPA Workshop on Human Language Technology*. Morgan Kaufmann, (1994).

[16] Dominique Perrin, 'Finite automata', *Handbook of Theoretical Computer Science*, 2–56, (1990).

[17] Emmanuel Roche, *Analyse Syntaxique Transformationelle du Français par Transducteurs et Lexique-Grammaire*, Ph.D. dissertation, Université Paris 7, January 1993.

[18] Emmanuel Roche, 'Parsing with finite-state transducers', in *Finite-State Devices for Natural Language Processing*, eds., Emmanuel Roche and Yves Schabes, MIT Press, (1996). Forthcoming.

[19] Max Silberztein, *Dictionnaires Electroniques et Analyse Lexicale du Français— Le Système INTEX*, Masson, 1993.