Left-to-Right Constraint-based Parsing of Head-final Languages

Zelal Güngördü
Department of Computer Engineering and Information Science
Bilkent University, Ankara 06533, Turkey
zelal@cs.bilkent.edu.tr

July 28, 1998

Abstract

We present a parsing algorithm for HPSG grammars that are specified in the form of a type hierarchy, with the constraints governing certain types imposed on the respective types in the hierarchy. The algorithm works directly on the representations provided by the HPSG formalism, making essential use of the selection features governed by the formalism. Parsing a string starts with an underspecified structure (assigned to that string), and proceeds by attaching every word of the input, from left to right, to that global structure, thereby dynamically changing it as the parse progresses. We propose certain strategies for a *head-final* language that guarantee the correct parse/parses with the least possible number of processing steps in most cases (and with minimal reanalysis in the remaining ones), which makes the algorithm particularly interesting for such languages.

1 Introduction

Constraint-based theories of grammar in general are highly lexicalized, meaning that they aim to represent as much of the linguistic information as possible in the lexicon. The presentation of Head-driven Phrase Structure Grammar (HPSG) by Pollard and Sag (1994) achieves this goal to a certain extent by the use of a type hierarchy and highly articulated lexical entries, but also leaves a good deal of information outside the lexicon, in the form of principles and phrase structure schemata. More recently, Sag (1997) has introduced an alternative view of certain concepts in the theory, which relies heavily on a type hierarchy, supporting a totally "lexicalized" characterization of HPSG grammars. We present a left-to-right parsing algorithm for such grammars that works directly on the representations provided by the HPSG formalism, combining bottom-up projection with top-down prediction in an essential way. The reader will notice (in the sections to come) that the algorithm exhibits a left-corner parsing behaviour, only that behaviour is expressed in terms of the selection information encoded in lexical entries rather than syntactic categories in phrase structure rules. Also, a concept of *selecting daughter* takes over the significance that the linguistic concept *head* bears in head-corner parsing (Kay, 1989).

Parsing a string in this approach always starts with an *underspecified* global structure, and proceeds by attaching every word in the input string, from left to right, to that structure, thereby gradually constraining the structure as the parse progresses. Once all the words in the input are consumed, the parser returns the global structure as its output, which – to be an acceptable one – is required to be *fully* specified.

Although the algorithm involves a certain degree of non-determinism, it is possible to adopt a number of strategies which in the case of a head-final language guarantee the correct parse/parses with the least possible number of processing steps in most cases (and with minimal reanalysis in the remaining ones). Therefore, we particularly propose the present algorithm for parsing head-final languages. Accordingly,

after a general description of the algorithm in Sections 3–6, we focus in Section 7 on parsing a head-final language, more specifically Turkish (also a "free" word order language), and provide examples from that language to illustrate how the algorithm works.

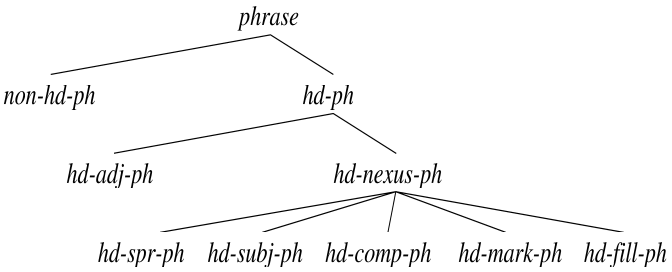
It is noted by van Noord (1997) that maintaining a chart in the case of constraint-based parsing often turns out to be highly costly (in terms of space). Note that the present algorithm does not make use of such a chart, yet parsing efficiency in certain cases could be improved, in a particular implementation, using *memoization* selectively (as proposed by van Noord (1997) for head-corner parsing); see Section 7.

2 The Formalism

The present work is based on a version of the HPSG formalism presented in Chapter 9 of Pollard and Sag (1994), slightly modified along the lines of Sag (1997). The hierarchy of constituent structures and the DTRS feature have been eliminated from the formalism. Instead, a feature geometry of phrasal signs has been devised that directly reflects constituent structures of phrases. This modification enables one to formalize the universal principles of HPSG (such as the Head Feature Principle, Valence Principle, and Semantics Principle) and also any language specific (parochial) principles, in the form of constraints imposed on certain phrase types in the hierarchy.

We assume the hierarchy of phrasal signs given in (1), which is a much simpler version of the one proposed by Sag (1997), but is nevertheless sufficient for our purposes. The Head Feature Principle, for example, can then be formulated as a constraint on phrases of type *headed-phrase (hd-ph)* as in (2). (following the formalization in (Sag, 1997)).

(1)



(2) Head Feature Principle:

$$\textit{hd-ph} \Rightarrow \begin{bmatrix} \text{HEAD} & \boxed{1} \\ \text{HD-DTR} & [\text{HEAD } \boxed{1}] \end{bmatrix}$$

This way, the grammar of a language needs only be specified in the form of a type hierarchy, with the constraints governing certain types imposed on the respective types in the hierarchy, including the inventory of words that belong to the various lexical types (Sag, 1997).

3 Use of Underspecification

The algorithm starts parsing a string always with an underspecified (global) structure (assigned to that string), and proceeds from left to right by attaching every word in the input string to STRUCT, as soon as that word is encountered and fully processed. It is a well-known fact that the possibility of left recursive structures in natural languages poses a fundamental problem for this kind of word by word parsing in general. (Examples of such structures include N′s with post-modifier PPs in English and German, for instance, and also embedded sentences with a sentence-initial complement clause in head-final languages such as Turkish.) Milward (1994) notes that the solution required is a way of encoding an infinite number of tree fragments, and presents an overview of different approaches in the literature to tackle the problem in that way.

To deal with the problem here we let the parser commit itself to decisions, concerning the attachment of newly encountered constituents, which it may *nonmonotonically* revise at subsequent stages of processing, if the need arises. One must however make sure that the nonmonotonicity embodied in the parser doesn't have any undesirable consequences on its soundness. To that end, one can take account of the fact that in HPSG any kind of *selection* is always realized via objects of type *synsem* that are structure-shared with the SYNSEM values of the daughters being selected. Thus, any nonmonotonic operation affecting the SYNSEM value of an object in the structure should be avoided, since it may override certain constraints imposed by the grammar on the structure at the previous stages of processing.[1] Moreover, one has to further ensure that there are no constraints imposed by the grammar on the type of the object to which the nonmonotonic operation applies that relates anything within the SYNSEM value of the object to a field affected by nonmonotonicity. For that, *only for processing purposes* we modify the type hierarchy in (1) introducing a new type called *hd-ph-pr* – with no linguistic significance – inserted between the types *phrase* and *hd-ph* (i.e. a subtype of *phrase* and a supertype of *hd-ph*).[2] This new type has the two appropriate features HD-DTR and NON-HD-DTRS[3] that take values of type *sign* and *list(sign)*, respectively. It is essential that there are no constraints in the grammar imposed on the type *hd-ph-pr* that relate any non-SYNSEM feature value to the SYNSEM value itself, or any other field within that value. The nonmonotonicity in processing is then limited to only non-SYNSEM fields of underspecified objects of type *hd-ph-pr* in STRUCT, to avoid overriding any constraints imposed on the structure by the grammar.

4 Phrase Specification/Construction

We present a totally transparent approach for parsing HPSG grammars, one that works directly on the representations provided by the HPSG formalism. To that end, it makes systematic use of the selection features governed by the formalism, such as the valence features SPR, SUBJ and COMPS, and the head selection features MOD and SPEC, in determining the type of phrase that a particular word (that is, the one just being processed) may be a part of.[4] During the processing of a single word, every *special* selection feature-value pair (cf. Section 6) in the lexical entry of that word *non-deterministically* triggers either of the following steps:

i) **Phrase specification:** The further specification of an underspecified phrase *hd-ph-pr* object in STRUCT, as an instance of one of the most specific phrase types (e.g. *hd-adj-ph, hd-spr-ph, hd-subj-ph*, etc) that is signified by the selection feature in question.

ii) **Phrase construction:** The construction of a new phrase object of the type (one of the most specific ones) signified by the selection feature in question, which is then attached to STRUCT as a daughter of one of the underspecified phrase *hd-ph-pr* objects.

Moreover, a phrase specification/construction step *may* further lead the parser to predict any yet unencountered daughter(s) of the phrase just specified/constructed, which is (are) being selected for via the selection feature initiating that step (as further discussed in Section 6).

[1] Although the selection of filler daughters is actually realized via objects of type *local* that are structure-shared with the LOCAL values of the selected filler daughters, that doesn't invalidate the present argument, since the LOCAL value of a sign is essentially a part (substructure) of its SYNSEM value.

[2] Here we deal with parsing headed phrases only. In other words, we do not deal with certain structures that are assumed to be non-headed in HPSG, such as coordinate structures.

[3] That is, in addition to the appropriate features it inherits from its supertypes, e.g. SYNSEM, PHON.

[4] In that respect, the present approach bears a certain resemblance to the work on compilation of HPSG grammars into feature-based TAGs by (Kasper et al., 1995), which also exploits the selection features in the HPSG formalism in projecting the lexical types of HPSG grammars to the elementary trees of the TAG formalism.

Given that phrases in natural languages may have structures with an arbitrary number of embeddings, one needs a way of keeping track of the underspecified phrase objects in STRUCT for the purposes of both steps mentioned above. To that end, we make use of a stack of phrases, PRED(ICTED)-PHRASES, whose elements are simply pointers to the underspecified substructures of STRUCT that are predicted by some already encountered daughter; e.g. complement daughters predicted by lexical heads, and head daughters predicted by specifiers, adjuncts or markers. Only one of those predicted daughters – the one on the top of the stack – is considered to be *active* at any given point in the course of the parse, meaning that the parser can only further specify that underspecified phrase (via a specification step), or attach a newly constructed phrase as a daughter of only that phrase (via a construction step).

A phrase specification step always leads the parser to pop the active phrase, which is now fully specified as an instance of one of the most specific phrase types, off the stack. In addition, a phrase specification/construction step *may* further lead the parser to push new underspecified phrases onto the stack that are predicted by the word just being processed (as mentioned above).

PRED-PHRASES, at the beginning of the parse, is initialized to contain a pointer to the global structure, STRUCT, the only attachment site available at that stage. At the end of the parse, PRED-PHRASES must be empty for the string to be grammatical, which guarantees that any predicted phrases in the course of the parse will indeed have been encountered by the end. It should be noted that any underspecified phrase during processing is pushed onto the stack, and that a phrase is popped off the stack only after (and as soon as) it is fully specified (i.e. constrained as an instance of one of the most specific phrase types). Consequently, requiring the stack to be empty guarantees that the initially underspecified structure assigned to the input string at the start will have been fully specified by the end of the parse. Also, although STRUCT itself may be popped off the stack at some point before the end of the parse, any change caused by specification/construction steps to the elements on the stack even after that point will have an immediate effect on STRUCT, since those elements are essentially pointers to certain substructures of STRUCT.

5 Parsing Algorithm

The main body of the parsing algorithm and the grammaticality principle are presented in (3) and (4), respectively. 'ND' stands for 'non-deterministically', and 'STACK (X)' is a function that returns a stack whose only element is a pointer to object X. Note that the non-determinism in step (3ci) is due to the possibility of lexical ambiguity in the language, and the one in step (3cii) is due to the choice the parser is supposed to make between a phrase specification step and a corresponding phrase construction step, as mentioned in Section 4. The algorithm, in its most general form, doesn't commit itself to any particular strategy to deal with the non-determinism introduced by these two steps (see however the discussion in Section 7).

(3) **Main Body:**

a. Constrain STRUCT as an object of type *hd-ph-pr*.
b. Initialize PRED-PHRASES to **STACK** (STRUCT).
c. For each WORD encountered do
   i. (ND) Fetch LEX for WORD from LEXICON.
   ii. (ND) Attach LEX to STRUCT.
d. Return STRUCT.

(4) **Grammaticality Principle:**

At the end of the parse (i.e. once all the words in the input are processed), PRED-PHRASES must be *empty*.

Although the main body itself is general, the attachment process, (3cii), varies for languages with different word order properties as a result of the fact that those languages employ different linear precedence constraints to account for the word order restrictions they exhibit. Accordingly, that process should be similar for languages with similar word order properties.

Following the notion of 'dynamics in algorithm development' introduced by Milward (1994), one can view the parse of a given input string using this algorithm in a dynamic way, as a sequence of states, where a pair of consecutive states represents a transition from the former state in the pair to the one following, by the complete processing of a single word in the input string (cf. step (3c) in the main body above). Each state is then composed of the values of the global structure, STRUCT, and the stack of predicted phrases, PRED-PHRASES, at a certain stage of the parse. In the initial state of the parse, STRUCT is only constrained as an object of type *hd-ph-pr*, and PRED-PHRASES only contains a pointer to STRUCT. In the final state, PRED-PHRASES must be empty for the string to be grammatical (and STRUCT then will have been constrained as an instance of one of the most specific phrase types, since that is the only way it may have been popped off the stack); cf. Section 4).

6 Attachment Process

As mentioned earlier, the parser benefits from the selection features in the formalism to determine the type of the phrase that a particular word with a particular selection feature-value pair may be a part of. So, for example, a *non-empty* SUBJ value in a given word's lexical entry indicates that a phrase headed by that word (i.e. a projection of the word) is to function as the head daughter in a head-subject phrase. Similarly, a *non-empty* SPR value signals (for a projection of the word) a head daughter role in a head-specifier phrase. In addition to these valence features, via which heads select for their arguments, the HPSG formalism also equips certain non-head daughters such as adjuncts, specifiers and markers with special features that enable them to select for the heads they are to modify or specify. Adjuncts, for instance, select for their heads via the MOD feature, for which they always have a value of type *synsem* that is to be structure-shared with the SYNSEM value of the head daughter in a head-adjunct phrase. (Likewise, specifiers and markers select for their heads via a *synsem*-valued SPEC feature.) Consequently, a *synsem*-valued MOD (SPEC) feature may be considered an indication of a non-head daughter role in a head-adjunct (head-specifier or a head-marker) phrase. The point to note is that, depending on the word order restrictions a certain language exhibits, a particular special selection feature-value pair X (such as the ones mentioned above) should lead the parser to *either* of the following two alternatives:

i) If the daughter bearing X is expected to follow the daughter that it selects for via X (according to the word order restrictions in the language), then, on encountering a word with X, the parser should infer that the daughter that is being selected for must have already been encountered, and hence no further prediction is necessary.

ii) If the daughter bearing X is expected to precede the daughter it selects for via X, then processing a word with X should lead the parser to predict the daughter that is selected for, by pushing it onto the stack of predicted phrases.

So, for example, in a head-final language where arguments and adjuncts always precede their heads, a *synsem*-valued MOD feature enables the parser to predict the head daughter at the time of processing the adjunct daughter. In the case of a *non-empty* SUBJ value, on the other hand, the parser infers that a phrase that has already been attached to STRUCT is to function as the subject of a projection of the word just being processed. One can then summarize the attachment process as follows:

(5) **Attachment Process:**

a. For each special selection value in LEX do
   i. (ND) Perform a specification/construction step.
b. Push any daughters predicted in (ai) onto the stack.

One must note that the order in which the selection features are considered in (5a) is significant. For instance, for some languages all headed phrases except head-complement phrases are constrained in the grammar to have a head daughter with an empty COMPS value, meaning the head must have already 'consumed' all its complements. Consequently, while parsing such languages (grammars), the parser should first consider the COMPS feature in step (5a). Note also that in case of two or more specification/construction steps called in (5a), each one of those works on the output of the one called just before, thereby recursively projecting the word (being processed) to several (embedded) phrase objects. Also, as mentioned earlier in Section 3, the parser is allowed during this process to non-monotonically re-attach certain phrases already attached as daughters of underspecified (*hd-ph-pr*) phrases in STRUCT, thereby moving those daughters further down to embedded phrases. Finally, any daughters predicted by specification/construction step(s) in (5a) are pushed onto the stack in (5b), in the reverse order as they are predicted, with the assumption that all phrase types specify continuous structures.

7 Implementational Issues

We have implemented an HPSG parser for Turkish, a "free" word order, head-final language, using the present algorithm, in the LIFE programming language (Aït-Kaci and Lincoln, 1988). LIFE provides the programmer with features from three different programming paradigms, namely functional programming, logic programming and object-oriented programming; e.g. functions, predicates, a Prolog-like resolution strategy, unification, memoization, an inheritance-based sorted feature system, multiple inheritance and constrained sorts (Aït-Kaci et al., 1994). The Turkish grammar embodied in the parser merely consists of a type (sort) hierarchy, with the universal and language specific principles imposed as constraints on certain phrase types in the hierarchy. To deal with the "free" nature of word order in Turkish, we assume a flat structure for Turkish sentences, in which all complements (including subjects) are treated in the same way.[5] Consequently, Turkish sentences are considered instances of *hd-comp-ph* rather than *hd-subj-ph*.

LIFE's (Prolog-like) depth-first resolution strategy with backtracking makes it seem natural, in such an implementation, always to give a specification step priority over a corresponding construction step. In fact, if one constrains the global structure, STRUCT, at the beginning of the parse, to be a *finite* verb projection, that strategy for Turkish almost always leads the parser to the correct parse/parses without any backtracking of a *successful* specification/construction step (and with minimal backtracking in the remaining cases; see below). This is a consequence of the fact that in all phrase types in Turkish except *hd-comp-ph* the selecting daughter (with a special selection feature-value pair) precedes the daughter that it selects for. And for head-complement phrases we adopt a strategy where consecutively encountered constituents are first attached as non-head daughters of the same clause, and are later re-attached (lowered) as non-head daughters of the embedded clauses once the respective heads of those clauses are encountered.

Let us now consider, as an example, the parse of (6), with a sentence-initial NP complement.

(6) Küçük çocuk uyu-yor.
little child sleep-PROG
'The little child is sleeping.'

[5] To that end, we make use of a lexical rule that applies to *non-base* verbs, removing the only element in the SUBJ list of the input entry, and placing it within the COMPS list of the output verb, thereby allowing that verb to select its subject via the COMPS feature, rather than SUBJ.

Assuming that STRUCT is constrained to be a finite verb projection (i.e. to have a verb[*fin*] HEAD value) at the start, the first word *'küçük'* in (6) would lead the parser to the parse state in (7),[6] where *'küçük'* is attached as the non-head daughter of a newly constructed head-adjunct phrase (by a construction step for head-adjunct phrases initiated by the *synsem* valued MOD feature in the lexical entry of *'küçük'*). That phrase is further attached as the non-head daughter of STRUCT (the then active phrase). Note also that the yet-to-come head-daughter, [2], of the head-adjunct phrase is pushed onto the stack, and that daughter is constrained as an NP at this stage of the parse (via a constraint on type *hd-adj-ph* in the grammar (type hierachy) that imposes a structure-sharing – indicated by [1] in (7) – between the MOD value of the non-head daughter of any *hd-adj-ph* object and the SYNSEM value of its head daughter).

After that, the second word *'çocuk'* would simply be attached, by a specification step, to the active phrase [2], leading to the parse state in (8). Note that [2] has now been further constrained as *hd-comp-ph*, and popped off the stack (by the specification step just mentioned).

Finally, the finite verb *'uyuyor'* is attached as the head daughter of STRUCT again by a specification step that further constrains STRUCT as *hd-comp-ph*, popping it off the stack, as shown in (9). Notice that the stack is now empty, satisfying the Grammaticality Principle, (4), for the output structure to be acceptable.

(7)

STRUCT

$$
\boxed{3}\begin{bmatrix}
\textit{hd-ph-pr} & & \\
\text{SYNSEM} & \left[\ \text{HEAD}\quad \textit{verb}[\textit{fin}]\ \right] & \\
\text{NON-HD-DTRS} & \left\langle \begin{bmatrix}
\textit{hd-adj-ph} & & \\
\text{SYNSEM} & \left[\ \text{HEAD}\quad \boxed{3}\ \textit{noun}\ \right] & \\
\text{NON-HD-DTRS} & \left\langle \begin{bmatrix}
\textit{hd-comp-ph} & \\
\text{PHON} & \langle \textit{küçük} \rangle \\
\text{SYNSEM} & \left[\ \text{MOD}\quad \boxed{1}\ \right] \\
\text{HD-DTR} & \textit{word}
\end{bmatrix} \right\rangle & \\
\text{HD-DTR} & \boxed{2}\begin{bmatrix}
\textit{hd-ph-pr} & \\
\text{SYNSEM} & \boxed{1}\left[\ \text{HEAD}\quad \boxed{3}\ \right]
\end{bmatrix} &
\end{bmatrix} \right\rangle \\
\text{HD-DTR} & \textit{sign} &
\end{bmatrix}
$$

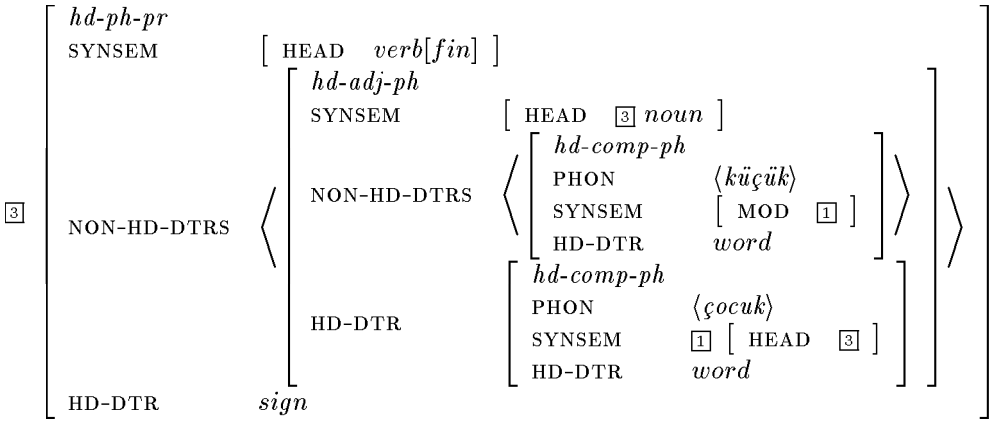
PRED-PHRASES

$$\langle\, \boxed{2}, \boxed{3}\, \rangle$$

[6] Concerning this parse state, and the ones to come, one should note the following: i) To improve the readibility – and also due to space limitations – we only show the feature values essential to the ongoing discussion. ii) The stack PRED-PHRASES is represented in the list notation with the top element on the stack on the left end of the list. iii) Re-entrancies not within a single feature structure should be viewed as pointers to the same linguistic object from an implementational point of view.

(8)

STRUCT

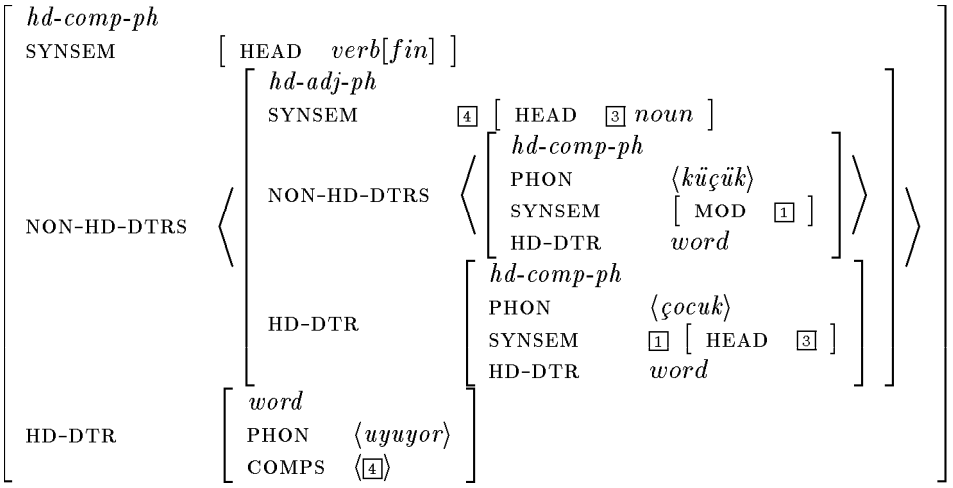


PRED-PHRASES

< 3 >

(9)

STRUCT



PRED-PHRASES

<>

One must note (about this parse) that constraining STRUCT as a finite verb projection at the start prevents the parser from applying a specification step for head-adjunct phrases while processing the first word *'küçük'* in (6) instead of the construction step mentioned earlier. If that specification step were allowed then STRUCT would be constrained as an NP of type *hd-adj-ph*, and the verb *'uyuyor'* would force the parser to backtrack all the way to processing *'küçük'*, to provide the correct parse.

However, in the case of a sentence-initial S[*fin*] complement, such as the one in (10),[7] even when STRUCT is constrained as before at the start the parser can still attach the embedded verb *'uyuyor'* as the head daughter of STRUCT by a specification step, which then leads to backtracking by the main verb *'sandım'*, but only to the point of processing the previous word *'uyuyor'*. A construction step then (instead of specification) constructs a new *hd-comp-ph* object headed by *'uyuyor'* re-attaching the sentence-initial NP as a non-head daughter of the newly constructed phrase, and further attaches that phrase as a non-head daughter of STRUCT. The main verb *'sandım'* can then be attached as the head-daughter of STRUCT, providing the correct parse.

[7]Note that Turkish is a pro-drop language and the first person singular subject in (10) has been dropped.

(10) [Küçük çocuk uyu-yor] san-dı-m.
little child sleep-PROG think-PAST-1SG
'I thought that the little child was sleeping.'

One can further improve the parsing efficiency in such cases using the memoization mechanism (provided by LIFE) selectively (van Noord, 1997), by making the parser memorize S[*fin*] structures only.

8 Conclusions

We have presented a left-to-right parsing algorithm for HPSG grammars that are specified in the form of a type hierarchy, with the constraints governing certain types imposed on the respective types in the hierarchy. The algorithm works directly on the representations provided by the HPSG formalism, combining bottom-up projection with top-down prediction in an essential way. We have proposed certain strategies for Turkish, which can further be readily adopted for other head-final languages such as Japanese and Korean, that guarantee the correct parse/parses with the least possible number of processing steps in most cases (and with minimal reanalysis in the remaining ones).

References

H. Aït-Kaci and P. Lincoln. 1988. Life: A natural language for natural language. Technical Report ACA-ST-074-88, Systems Technology Laboratory, Austin, Tex.

H. Aït-Kaci, B. Dumant, R. Meyer, and P. V. Roy, 1994. *The Wild LIFE Handbook.* Digital-Paris Research Laboratory, prepublication edition.

R. Kasper, B. Kiefer, K. Netter, and K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. In *Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics*, Cambridge, Mass.

M. Kay. 1989. Head-driven parsing. In *Proceedings of the International Workshop on Parsing Technology*, Pittsburgh, Pa.

D. Milward. 1994. Dynamic dependency grammar. *Linguistics and Philosophy*, 17:561 – 605.

C. Pollard and I. A. Sag. 1994. *Head-Driven Phrase Structure Grammar.* CSLI Lecture Notes. CSLI and University of Chicago Press, Stanford, Ca. and Chicago, Ill.

I. A. Sag. 1997. English relative clause constructions. *Journal of Linguistics*, 33(2):431 – 484.

G. van Noord. 1997. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3):425 – 456.