93338

GENERATION OF TURKISH SURFACE FORM FROM A MORPHEMIC LEXICON

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

BURCU KARAGÖL-AYAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

JULY 2000

Approval of the Graduate School of Natural and Applied Sciences.

[illegible]

Prof. Dr. Tayfur Öztürk
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Fatoş Yarman Vural
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Assoc. Prof. Dr. Cem Bozşahin
Supervisor

Examining Committee Members

Assist. Prof. Dr. Halit Oğuztüzün	[illegible]
Assoc. Prof. Dr. Cem Bozşahin	[illegible]
Assoc. Prof. Dr. Ferda Alpaslan	[illegible]
Assoc. Prof. Dr. Deniz Zeyrek	[illegible]
Dr. Ayşenur Birtürk	[illegible]

ABSTRACT

GENERATION OF TURKISH SURFACE FORM FROM A MORPHEMIC LEXICON

Karagöl-Ayan, Burcu
MS., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Cem Bozşahin

July 2000, 65 pages

In this thesis, our goal is to generate Turkish surface forms from Predicate-Argument Structure (PAS) using a morpheme-based lexicon. The system uses a model which integrates inflectional morphology and syntax in which the building blocks are morphemes. The algorithm is based on bottom-up use of the category-semantics pairing coming from the lexicon. The generator presented in this thesis is capable of partial phrase formation. We discuss design and implementation aspects of the generator.

Keywords: Natural Language Processing, Categorial Grammar, Generation, Predicate-Argument Structure, Morpheme, Lexicon

ÖZ

TÜRKÇE YÜZEY BİÇİMİNİN BİÇİMBİRİMSEL SÖZLÜKBİLGİSİNDEN ÜRETİMİ

Karagöl-Ayan, Burcu
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Cem Bozşahin

Temmuz 2000, 65 sayfa

Bu tez çalışmasında amacımız biçimbirime dayalı sözlükbilgisi kullanarak Yüklem-Öğe Yapısından (YÖY) Türkçe yüzey biçimleri üretmektir. Sistem, biçimbirimlerinin yapı taşı olduğu, çekimsel biçim bilgisi ve cümle bilgisini birleştiren bir model kullanmaktadır. Algoritma sözlükbilgisinden gelen ulam-anlam eşleşmesinin aşağıdan yukarıya doğru kullanılmasına dayanmaktadır. Bu tez çalışmasında sunulan üreteç kısmi söz öbekleri de oluşturabilmektedir. Üretecin tasarım ve uygulama yönleri irdelenmiştir.

Anahtar Kelimeler: Doğal Dil İşleme, Ulamsal Dilbilgisi, Üretim, Yüklem-Öğe Yapısı, Biçimbirim, Sözlükbilgisi

To my family,

my love Fazıl,

and...

the wonders of mind and nature...

ACKNOWLEDGMENTS

Thanks to my advisor Cem Bozşahin for assigning me this topic. Also thanks for his guidance throughout this thesis, and his fast reviews and valuable corrections to the manuscripts.

Thanks to my friends for their support and for being my friend.

Thanks to my family for their love, moral support and confidence in me. Thanks to my father Recep, my brother Ahmet and my sister Ahsen for being part of my family. Thanks to my mother Melek Nur for her everyday telephone calls during the writing of this thesis. Special thanks to my sister Tuba for sparing her time to me, and her efforts to entartain me whenever I need during the writing of this thesis.

And mostly thanks to my husband, my love Fazıl without whom this thesis would not have been finished. Thanks for his interest in this study by discussing my ideas and giving valuable comments. Thanks for his patience, encouragement, trust, and support under any circumstances. And mostly thanks for his love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Natural Language Processing (NLP) deals with computation of natural languages using computers. This computation can be (i) deriving meaning and structure from a surface form (analysis), or (ii) production of surface form from structured meaning (generation).

Parsing and generation are two important processes in computational linguistics. Parsing is the recovering of a meaning encoding from a given natural language expression, whereas generation is producing a well-formed natural language expression from an encoding of its meaning. During generation process, a generation grammar and a generation lexicon are used.

In this study, our aim is to generate surface forms for Turkish. Turkish is an agglutinating language with rich morphology. The division of syntax and morphology is especially difficult for languages like Turkish. Hence, it is a reasonable approach to integrate inflectional morphology and syntactic processes. The model used in this study is a morphosyntactic one; morphemes are the building blocks. The term morphosyntax in this study refers to those aspects of morphology and syntax that collectively contribute to grammatical meaning composition. There is no distinction between phrase formation and word inflection.

Taking the morphemes as the unit of representation is especially suitable for Turkish. In this way, the correct semantic bracketing of a phrase, when a morpheme has a phrasal scope, derives straightforwardly without rebracketing[1]. For instance, in the

[1] Rebracketing is changing the scope relations due to a mismatch, in our case, mismatch with semantics, e.g., PLU(green bus) vs. green(PLU bus)

following sentence (1) the scope of the case marker is shown. This scope is motivated by the semantics of the utterance. In a morpheme-based approach, this derivation is straightforwardly obtained, while in a word-based approach, getting this bracketing requires further processes, such as rebracketing, and this will make the morphology-syntax-semantics interface nontransparent.

(1)	*Adam*	*[kadın-ın*	*gel-diğ-i*	*]-ni*	*gör-dü*
	Man	woman-GEN.3	come-SUB-POSS.3s	-ACC2	see-TENSE

'The man saw that the woman was coming.' = *see(man,(come woman))*

The motivation behind this study is the framework proposed by Bozşahin (1999) and the parser implemented using this model (Bozşahin, 1998). The aim of the study is to generate Turkish surface forms from a structured meaning representation called the Predicate Argument Structure (PAS) using a categorial morphosyntactic framework which is based on Combinatory Categorial Grammar (CCG) (Steedman, 1985; Steedman, 1987; Steedman, 1988; Steedman, 1996). The parser takes Turkish surface forms as input and derives the PAS. The generator does the reverse of this operation: it takes the PAS and the final category of the surface form as input and produces corresponding Turkish surface form. The output is generated using the morphemes as the main building blocks; this is a morpheme-based generation rather than a word-based generation. Taking the morphemes as the unit of representation results in the transparency of syntax, inflectional morphology, and semantics.

The generation algorithm used in this study is an adaptation of the semantic-head-driven generation algorithm (Shieber et al., 1989; Shieber et al., 1990), which takes advantage of both top-down and bottom-up information and suitable for lexicalist approaches.

The thesis is organized as follows. Chapter 2 presents some background. Relevant characteristics of Turkish, CCG, and generation algorithms are described in this chapter. The categorial model is explained in Chapter 3. The morpheme-based lexicon used in this study is described in detail in this chapter. In Chapter 4, the highlights of the morphosyntactic generation of Turkish is presented. The process of generation is described with examples. We then present a discussion comparing our approach with similar work on Turkish generation.

CHAPTER 2

BACKGROUND

In this chapter, we review some of the topics related to this study. In the next section, an overview of some aspects of Turkish language is given and morphophonemic rules are described. The aspects of Combinatory Categorial Grammar (CCG), the formalism used in this study, are briefly described in Section 2.2. An overview of the generation algorithms is given in the last section.

2.1 Turkish

Turkish belongs to the Altaic branch of the Ural-Altaic family of languages. It is an agglutinating language, that is, grammatical features can be indicated by adding various inflectional morphemes[1] to words. It has complex agglutinative word forms with productive inflectional and derivational morphological processes. We describe some aspects of Turkish morphology that are particularly relevant to this work.

There are seven cases: nominative, accusative, dative/allative, locative, ablative, genitive, and comitative/instrumental. Nominative case is phonologically null. Number is marked by a plural suffix. Nouns can also take the relative suffix. The same suffixes that apply to common nouns apply to proper nouns, pronouns, question words, and adjectives[2] as well. Verb stems can take suffixes for voice, aspect, negation, tense, question, etc. Subject-verb agreement is marked by person and number morphemes on

[1] In Turkish, affixes are generally in the form of suffixes, only a few prefixes exist.

[2] Adjective and noun distinction in Turkish is difficult. Most adjectives can be used as nouns, and nouns can perform the function of an adjective as noun modifier in noun-noun groups.

a. Noun inflections

Noun stem	Plural	Possessive	Case	Relative

b. Verb inflections

Verb stem	Voice	Negative	Compound verb	Main tense	Question	Second tense	Person

Figure 2.1: Noun and Verb Inflection Orders

the verb. The third person singular morpheme is phonologically null, and third person plural marker can be dropped.

The order of Turkish noun inflections are given in Figure 2.1a and the order of Turkish verb inflections are given in Figure 2.1b (Oflazer, Göçmen, and Bozşahin, 1995). For instance, a Turkish noun cannot take the number affix if it is already case-marked. Therefore, the phrase in (2a) is possible in Turkish, while the one in (2b) is not.

(2) a. *dağ-lar-a*
mountain-PLU-DAT
'to the mountains'

b. **dağ-a-lar*

A noun stem to which a relative suffix is affixed behaves like a nominal root; it can receive nominal inflections in Figure 2.1a in that order again (3).

(3) *dağ-da-ki-ler-e*
mountain-LOC-NREL-PLU-DAT
'to the ones at the mountain'

After derivational morphemes that may change the part-of-speech of a word, the stem takes the inflectional suffixes that are applied to its final part-of-speech category. For instance, a nominal root can become a verb via a derivational suffix and then takes verbal inflections.

In genitive noun groups, which can be nested, the pronominal possessor and the possessee agree in person and number (4). Pronominal possessors of possessive nouns may be omitted (5).

(4) *ben-im* *çocuğ-um-un* *arkadaş-ı*
man-GEN.1 child-POSS.1s-GEN.3 friend-POSS.3s
'the friend of my child'

(5) a. *ben-im* *ev-im*
I-GEN.1 house-POSS.1s
'my house'

b. *ev-im*

Turkish subordinate clauses are nominalizations: they are case-marked like noun phrases and have subjects marked in the genitive. Nominalization is evident from nominal inflection on the subordinate verb (6), but it also subcategorizes for arguments that are case-marked, just like a matrix verb's arguments. The subordinate verb also gets a subject-agreement marker that is overt even for third person singular subjects, and is assigned case like NP arguments (6).

(6) a. *Zeynep kedi-nin süt-ü iç-tiğ-i-ni söyle-di*
Zeynep cat-AGR milk-ACC drink-SUB-POSS.3s-ACC2 say-TENSE
'Zeynep said that the cat drank the milk.'

b. *Zeynep ben-im süt-ü iç-tiğ-im-i söyle-di*
Zeynep I-AGR milk-ACC drink-SUB-POSS.1s-ACC2 say-TENSE
'Zeynep said that I drank the milk.'

Turkish is a pro-drop language. Since the subject-verb agreement morphology on the verb gives the information concerning the person and number of the subject, the subject may be omitted if it is a pronoun. Turkish sentences within a discourse often have null subjects. In the case of third person plural, the person morpheme can be omitted if person information can be obtained from the subject, that is if the subject pronoun is not dropped. Although Turkish verbs are not marked for object-agreement or with object clitics, objects can also be dropped in Turkish. Null objects are quite commonly used instead of overt pronouns in discourse contexts where an antecedent to the null object can be easily found (Ümit Turan, 1995).

Table 2.1: Percentage of different word orders (500 adult, 100 child utterences)

Order	Children	Adult Speech
SOV	46%	48%
OSV	7%	8%
SVO	17%	25%
OVS	20%	13%
VSO	10%	6%
VOS	0%	0%

Turkish is a head-final language; modifiers/specifiers always precede the modified/specified. For example, adjectives always come before the nouns they modify. Adpositions follow their object NP.

Although subject-object-verb (SOV) is the most commonly used word order of a simple transitive Turkish sentence, all six permutations are grammatical. Since Turkish has overt case marking, this allows one to distinguish the subject and object of a sentence. The percentages of different word order usages in Turkish that Slobin provided (1978) are given in Table 2.1.

The propositional interpretation assigned to all six permutations of the transitive sentence in (7) is *read(mehmet,book)*. However, each word order conveys a different discourse meaning appropriate to a specific discourse situation.

(7) a. *Mehmet kitab-ı oku-du*
Mehmet book-ACC read-TENSE
'Mehmet read the book.'

b. *Kitabı Mehmet okudu*

c. *Mehmet okudu kitabı*

d. *Kitabı okudu Mehmet*

e. *okudu Mehmet kitabı*

f. *okudu kitabı Mehmet*

It is the context that determines the word order in Turkish. Since the order is pragmatically controlled, one would get a different reading for each variation in the order. Information conveyed through intonation, stress or clefting in fixed word order languages like English is expressed in Turkish by changing the word order. In (8), different readings of a sentence with the same propositional interpretation *give(Ahmet,man,money)* is shown. Each one has a different translation in English.

(8) a. *Ahmet para-yı adam-a ver-di*
Ahmet money-ACC man-DAT give-TENSE
'Ahmet gave the money to the man.'

b. *Ahmet adam-a para-yı ver-di*
'It is the money that Ahmet gave to the man.'

c. *Para-yı Ahmet ver-di adam-a*
'It is Ahmet who gave the money to the man.'

d. *Ver-di para-yı Ahmet adam-a*
'Ahmet did give the money to the man.'

Erguvanlı (1979) defined the sentence initial position as the *topic*, the immediately preverbal position as the *focus*, and the postverbal positions as *background* information. Turkish speakers tend to first place the information that links the sentence to the previous context (topic), then the important and/or new information immediately before the verb (focus), and the information that is not really needed but may help the hearer to understand the sentence better (background), after the verb.

Although word order is relatively free in Turkish, there are syntactic restrictions. For example, relativization must be head-final, and embedded clauses and subordinate clauses are strictly verb-final. Turkish direct objects are normally marked accusative. However, they can also occur without any case marking. When the direct objects occur in a sentence without case-marking, the OSV word order is not licensed with the nonreferential objects (9c) (Bozşahin, 1999).

(9) a. *Çocuk kitab-ı oku-du*
Child book-ACC read-TENSE
'The child read the book.'

b. *Çocuk kitap oku-du*

c. **Kitap çocuk oku-du*

Table 2.2: Turkish Vowels

	Low	High	Low	High
Back	a	ı	o	u
Front	e	i	ö	ü

Table 2.3: Turkish Consonants

	Bilabial	Labio-dental	Dental Alveolar	Palato-Alveolar	Palatal	Velar	Glottal
Stop-voiceless	p		t	ç		k	
Stop-voiced	b		d	c		g	
Fricative-voiceless		f	s	ş			
Fricative-voiced		v	z	j			
Nasal	m		n				
Liquid lateral			l				
Liquid nonlateral			r				
Glide					y	ğ	h

Morphophonemics of Turkish

The features of the 8 vowels and 21 consonants in Turkish are shown in Table 2.2 and Table 2.3 (Oflazer, Göçmen, and Bozşahin, 1995).

Phonological alternations in morphemes are defined as *morphophonemic rules.* Vowels in the affixes and the consonants in roots, and affixes undergo certain modifications, and may sometimes be deleted under conditions defined in these rules.

Vowel harmony rule in Turkish forces the vowels in the suffixes agree with the last vowel in the stems on backness and roundness.

Some suffixes have buffer phonemes. For example, the suffix *-(y)lA*[3] 'with' indicates that the phoneme *y* is a buffer, and it may drop in some cases. *A* is realized as an *a* or *e* according to vowel harmony.

Turkish words do not end with voiced stops (*b, c, d, g*). Their voiceless stop counterparts (*p, ç, t, k*) appear at the end of a word.

2.2 Combinatory Categorial Grammar

Although the original formulation of Categorial Grammar (CG) reaches as far as 1920's, the idea of employing it in the analysis of natural language was first adopted in the

[3] Capital letters represent meta-phonemes which will be explained in Section 3.3.

1960s. The reason for this renewed interest was that categorial grammar lends itself naturally to the kind of semantic types which can be put in correspondence with syntactic types. Bar-Hillel, Gaifman, and Shamir (1960) proved that a pure categorial grammar is weakly equivalent to a context-free grammar.

CGs are lexicalist formalisms, that is most of the information is put into the lexicon instead of having them in the grammar. A pure categorial grammar has the following four characteristics:

a) There is a finite set of *basic categories.*

b) There is a small set of *syntactic rules* describing the operation of a functor on an argument.

c) A countably infinite set of *derived categories* can be constructed using the syntactic rules.

d) Every lexical element is assigned a category.

Below is a very simple example of a pure categorial grammar with only backward directionality:

i) The basic categories are N (for 'noun') and S (for 'sentence').

ii) The syntactic rule is: if α is an expression of category A, and β is an expression of category $(B\backslash A)$, then $\alpha\beta$ is an expression of category B.

iii) The derived categories may be obtained as follows: If A and B are categories, then $(A\backslash B)$ is also a category.

iv) The categories of the lexicons are:

   Jack $:= N$

   speaks $:= S\backslash N$

   quietly $:= ((S\backslash N)\backslash(S\backslash N))$

$(S\backslash N)$ means that this expression looks for an N on its left to become S. According to this primitive categorial grammar, the expressions in (10) are of category S, in other words they are sentences (Steedman, 1996).

(10) a. *Jack speaks*

$$\dfrac{\overline{N} \quad \overline{S\backslash N}}{S}$$

b. *Jack speaks quietly*

$$\begin{array}{ccc} & & \\ \overline{N} & \overline{S\backslash N} & \overline{(S\backslash N)\backslash(S\backslash N)} \\ \cline{2-3} & S\backslash N & \\ \hline \multicolumn{3}{c}{S} \end{array}$$

In general, then, a categorial grammar enables us to determine whether the result of combining expressions of any given categories is itself a grammatical expression, and if so, to determine its category. Therefore, a categorial grammar provides an automatic procedure for determining which expressions are sentences and which are not.

A unidirectional categorial grammar is not strong enough to cover natural languages; *bidirectional* categorial grammars are used for this purpose. In bidirectional categorial grammars, there are two operators and two syntactic rules, which both deal with the same syntactic operation, namely, *application* (Steedman, 1996):

(11) Forward Application (>): $X{:}\, fx / Y{:}\, x \;\; Y{:}\, a \Rightarrow X{:}\, fa$

Backward Application (<): $Y{:}\, a \;\; X{:}\, fx \backslash Y{:}\, x \Rightarrow X{:}\, fa$

Categories also include semantic interpretations as shown above after colons. Here interpretations are in the form of *predicate-argument structures* (PAS). We will describe the PAS in Section 3.1 in more detail. As it can be seen from the rules, unification is used to accomplish syntax-semantics pairing. Intuitively, unification is an operation which amalgamates compatible terms or expressions and fails to amalgamate incompatible ones. In unification-based formalisms, X, Y, etc. in rules can be regarded as variables over categories which can be instantiated or given values by unificaton with categories like *S, NP,* or *S/NP.*

Combinatory Categorial Grammar, CCG, is an extension of CG (Steedman, 1985; Steedman, 1987; Steedman, 1988; Steedman, 1996). While the only rules in CG are application rules (11), CCG includes composition (12a-b), and type-raising (12c) as well. With these extensions, CCGs are equivalent to mildly context-sensitive languages (Joshi, 1985) which are more powerful than context-free languages.

(12) a. Forward Composition ($\mathsf{B}_{>}$): $X{:}\, fy / Y{:}\, y \;\; Y{:}\, gx / Z{:}\, x \Rightarrow X/Z{:}\, f(gx)$

Backward Composition ($\mathsf{B}_{<}$): $Y{:}\, gx \backslash Z{:}\, x \;\; X{:}\, fy \backslash Y{:}\, y \Rightarrow X\backslash Z{:}\, f(gx)$

b. Forward Crossing Composition ($\mathsf{B}_{\times >}$): $X{:}\, fy / Y{:}\, y \;\; Y{:}\, gx \backslash Z{:}\, x \Rightarrow X\backslash Z{:}\, f(gx)$

Backward Crossing Composition ($\mathsf{B}_{\times <}$): $Y{:}\,gx/Z{:}\,x \;\; X{:}\,fy\backslash Y{:}\,y \Rightarrow X/Z{:}\,f(gx)$

c. Type Raising (T): $X{:}\,a \Rightarrow T{:}\,fa/(T{:}\,fa\backslash X{:}\,a)$
or $T{:}\,fa\backslash(T{:}\,fa/X{:}\,a)$

Composition rules combine two function categories, syntactically and semantically, together. Type-raising turns an argument, such as an *NP*, into a functor that looks for a functor looking for *NP*. Type-raising rules are called order-preserving rules by Dowty (1988) because they preserve word order and dominance relationships. Composition and type-raising give extra power and complexity to CCG.

Handling some long-distance dependencies such as wh-questions and relative clause formations in English is possible by the use of type-raising and composition. Type-raising, in combination with the composition rules, can produce constituents that are nontraditional. For example, when we consider the English sentence "*Poirot solves mysteries*", the bracketing of a traditional grammar will be as in (13a), whereas both bracketings in (13) are possible with CCG. The bracketing in (13b) is a result of the composition of a type-raised subject with the verb before it combines with the object.

(13) a. [*Poirot* [*solves mysteries*]$_{S\backslash NP}$]

b. [[*Poirot solves*]$_{S/NP}$ *mysteries*]

Such nontraditional constituents are crucial for CCG. When they are allowed, handling coordination and extraction in English becomes possible. For (14), the nontraditional constituent [*Poirot solves*]$_{S/NP}$ is necessary.

(14) a. **Coordination:** [*Poirot solves*]$_{S/NP}$ *and* [*Hastings writes*]$_{S/NP}$ *murder cases.*

b. **Extraction:** *Hastings writes the mysteries that* [*Poirot solves*]$_{S/NP}$.

However, type-raising and composition cause the *spurious ambiguity* problem (Wittenburg, 1987), that is, during parsing multiple analyses with the same semantic interpretation may occur. Because of the non-standard constituents of CCG, even simple sentences have many semantically equivalent parses. For example, the sentence '*Poirot solves mysteries*' has two parses as shown in (13). These "extra" parses increase exponentially for longer sentences. Generating all parses in this sense is inefficient. Several solutions have been proposed to handle this problem. The solution used in this framework will be given in Section 3.3.

CCG is used in this study for several reasons: First of all, CCGs provide a compositional and flexible surface structures, which allow syntactic constituents to easily correspond with units of information structure. Secondly, CCGs are lexicalist formalisms. This improves the processing efficiency since we only have to look at the local information associated with each word in the input sentence, instead of all the information in the grammar. Thirdly, the notion of lexicon in CCG is neutral to the sound-meaning representation: any meaning-bearing element such as morpheme, word, or phrase can be in the lexicon. One extreme example is having a lexical entry for *Poirot solves*, which, as a non-standard constituent with the category S/NP, is fully interpretable with the semantics $\lambda x.solves'x\ poirot'$. In phrase structure formalisms, this cannot be a lexical entry because lexical insertion presumes standard constituents. The other extreme would be a case inflection as a lexical entry, such as the locative *-da* with the category $NP_{loc}\backslash N$ and semantics $\lambda x.at'x$. Fourthly, the same grammar and lexicon can be used for both parsing and generation. Furthermore, CCGs are non-transformational grammars that do not posit empty categories or movement rules (Hoffman, 1995). Transformations greatly increase the power of grammar which may affect its computational efficiency. Moreover, CCGs are mildly context-sensitive. Shieber (1985) showed that competence grammars for natural language must be more powerful than context-free grammars. The class of mildly context-sensitive grammars are formally adequate for natural languages while still being polynomially parsable (Weir and Joshi, 1988). Finally, CCGs are computationally efficient (i.e. polynomially parsable (Vijay-Shanker and Weir, 1993)).

2.3 Generation Algorithms

In most natural languge generation systems there are two different parts. The first one is the *strategic* part, and the other is the *tactical* part. Strategic part is about 'deciding what to say' whereas tactical part deals with 'how to say it' (van Noord, 1990). In a bidirectional grammar, relations between strings and a semantic representation, which is sometimes called the *logical form*, are defined. Parsing is the determination of the corresponding logical form(s) for a given string, whereas (tactical) generation is the determination of the corresponding string(s) for a given logical form. Therefore, we can say that parsing and generation are reverse processes.

Generation is not a new problem in computational linguistics. Early approaches to tactical generation problem were usually top-down methods. However, top-down generators have significant problems, which are discussed in Shieber et al. (1989), Shieber et al. (1990), and van Noord (1990) in detail. The principal problem in top-down generators is the left-recursion. The order in which the nonterminals are expanded can be important in such generators since the order of processing is always left-to-right. Therefore, top-down generators may fail to terminate on certain grammars. Lexical information cannot be used in top-down generators.

Shieber (1988) proposed a chart-based generator in which rules are applied in a bottom-up fashion, and results are kept on an Earley type chart. However, this generator has a strong requirement: semantic monotonicity, that is, the logical form of every subphrase must subsume some part of the input logical form. Moreover, it has processing overhead because of subsumption checks. The nondeterministic style of processing is another disadvantage of this generator.

Shortly, top-down generators have left-recursion problem, and chart-based bottom-up generators have strong restrictions on the grammars, and they are rather inefficient in processing.

Semantic Head-Driven Generation

Shieber et al. (1989; 1990) proposed a semantic head-driven generation algorithm. This algorithm is a significant improvement over the top-down methods and the previous bottom-up methods based on Earley deduction.

The semantic head-driven generation algorithm consists of two parts. In the *prediction* part, a lexical entry is predicted, and in the *connection* part, a parse tree which is matching the top goal and starting from the predicted lexical entry is built.

In semantic head-driven generation the order of processing is not left-to-right but always starts with the head of a construction. The logical form of the head is known during the prediction step of the algorithm. This is the top-down information. Apart from this top-down part, the algorithm uses the lexical information which is the bottom-up information. Therefore, semantic head-driven generators work in two ways: the semantic input, which is top-down, and the lexical information, which is bottom-up.

As van Noord (1990) suggested, this property of semantic head-driven generators are especially useful and appropriate for lexicalist grammars. Moreover, control from two different sources yields acceptable performance in generation process. In addition, the bottom-up approach does not suffer from left-recursion non-termination which is the major drawback of top-down generators.

There are several applications of semantic head-driven generation algorithm. A DCG implementation of the approach is discussed in Shieber et al. (1989), Shieber et al. (1990). van Noord's BUG (Bottom-Up Generator) (1989; 1990), an experimental machine translation system for translating international news items of Teletext which uses a Prolog version of PATR-II, is a successful application of the method. Calder et al. (1989) use a generation algorithm for Unification Categorial grammar (UCG) that appears to be a special case of head-driven bottom-up generation algorithm. Hoffman (1994) uses a word-based generation algorithm for Multi-set CCG which is an adaption of head-driven bottom-up generation.

CHAPTER 3

THE CATEGORIAL MODEL

In this chapter, first the characteristics of the PAS is explained. Then, the framework is described. In the last section, the lexicon used in the study is described in detail.

3.1 Predicate-Argument Structure

The Predicate-Argument Structure (PAS) is the only level of representation in CCG. It is the starting point of mapping of semantic interpretation to surface form in this study. The Principle of Combinatory Transparency, defined by Steedman (1996), indicates that the syntactic form of CCG rules entirely determines their semantics. Therefore, there is no way that these rules can access and change PAS. PAS is formed from the syntactic derivation.

The first element in PAS is the predicate, and the following are the arguments. The PAS reflects the order of the arguments between themselves; the last argument is the primary term, on its left is the secondary term, etc. (Bozşahin, 1998). The equivalent binary tree of this representation is shown in Figure 3.1b. This representation is left-associative to preserve dominance relationships and binding conditions on the arguments.

This representation does not reflect the surface order of the constituents. The verb is the predicate and the subject is the primary term. Word order is defined solely by the directional slashes in CCG.

In order to make explicit the place of an argument in the PAS, 'argument index' or

a. *Predicate... < Secondary Term >< Primary Term >*

b.

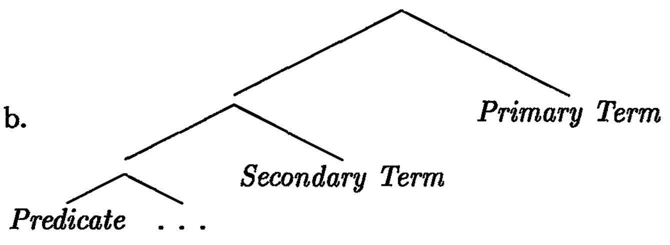


Figure 3.1: Predicate-Argument Structure

genotype index is used in this study. Therefore primary argument, which is the subject of the sentence, is indicated as NP_1, and the direct object as NP_2. The representation of the ditransitive verb *give* in this format is (Bozşahin, 1999):

(15) $give := (S{:}give\ xyz \backslash NP_1{:}z)/NP_2{:}y/NP_3{:}x$

Another concept is binding of reflexives and related anaphors. Steedman (1996) defines the true bounded anaphors as syntactically identical to other *NP*s; they have an additional ANA[1] feature. In this way, binding can be represented at the level of predicate-argument structure. For example, the usage of this binding feature will give another lexical category (16b) in addition to its normal lexical category (16a) for the transitive verb *see*. This second category can be used in sentences such as '*I see myself*.

(16) a. $see := (S{:}see\ xy \backslash NP{:}y)/NP{:}x$

b. $see := (S{:}g\ see\ (ANA\ y)\ y\ \backslash NP{:}y)/NP{:}g$

One of the essential use of binding and *ANA* is in relativization. We do not give the details on this subject, but their use is described in Section 3.3.1.9.

3.2 Categorial Framework

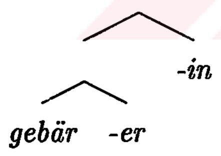
This study is based on the categorial framework proposed by Bozşahin (1998; 1999; 1995). In this model, morphology and syntax are treated not as separate components, but as a co-extensive domain. The morphological and syntactical processes are integrated, and semantic composition is performed in parallel to these. The semantic

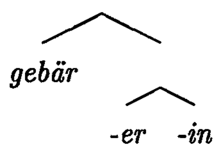
[1] Constants in the PAS are represented in capital letters.

representation is the predicate-argument structure. The idea behind this model is to deal with the difficulty of separating morphology and syntax in agglutinating languages, and the problems caused by processing them separately.

In Turkish, grammatical features can be indicated either by syntactical or morphological means. The grammatical features that are realized by words in a language such as English can be indicated by bound morphemes in Turkish. For instance, case marking is a morphologically marked function, whereas indirect objects are syntactically marked. When this morphosyntactic model is used the distinction between inflectional morphological and syntactical processes no longer exists. This categorial framework is morpheme-based, that is morpheme is the unit of representation in the lexicon and it has the same lexical representation as the word. The format of the lexicon is described in detail in Section 3.3.

Semantic bracketing mismatches is an important issue. The problem in other languages has been pointed out by e.g. Carpenter (1997), Williams (1981). Moortgart (1988) gives example from German. For the German word *gebärerin* 'female person who gives birth', the correct meaning is derived if the suffixes *-er* (the suffix that turns verbs into nouns) and *-in* (the suffix that forms feminine nouns) form a composite affix, and then attached to the word *gebär* (17b).

(17)a. *


b.


An example from Turkish is (Bozşahin, 1999):

(18) a. *otobüs bilet-ler-i*
bus ticket-PLU-COMP
'bus tickets' =(PLU(COMP ticket bus))

b. **otobüs bilet-i-ler*

In the example above, the nominal compounding morpheme *-i* should come before the plural morpheme *-ler* in order to get the correct semantics. However, as mentioned in Section 2.1, plural morpheme must attach to nouns before the other morphemes (18b). Hence, the predicate-argument structure and the surface form conflict in this phrase. Bozşahin (1999) proposes a solution to this inconsistency. Pluralization and compounding are composed into one morpheme as plural compound which has the same properties of a compound morpheme. Therefore, the morpheme *-leri* should be treated as a composite lexical marker of pluralization and compounding, and the phrase in (18) should be interpreted as (19). In this way, the bracketing problem disappears.

(19) *otobüs bilet-leri*

bus	ticket-COMP.PLU

'bus tickets' = (PLU(COMP ticket bus))

When this model is used, the derivation of ill-formed semantics is prevented since there is no distinction between inflectional morphology and syntax. However, new scope problems come up. Different semantic forms may be derived for one surface form:

(20) *kırmızı panjur-lu ev*

red	shutter-ADJ	house

For the example above, where a nominal derivational morpheme is used, two different compositions are possible. These are given in Figure (3.2). Both derivations are reasonable, and both semantic forms are meaningful. While in the first derivation the scope of the morpheme *-lu* is the single word *panjur*, in the second derivation its scope is the complex *kırmızı panjur*. These two interpretations can be accounted for as in (21).

(21) a. *kırmızı panjur -lu ev*

```
 ─────   ───   ─────────   ──
 N/N     N     (N/N)\N     N
         ─────────────<
              N/N
         ──────────────────────>
                    N
 ──────────────────────────────>
            N
```

'the red house with shutter'

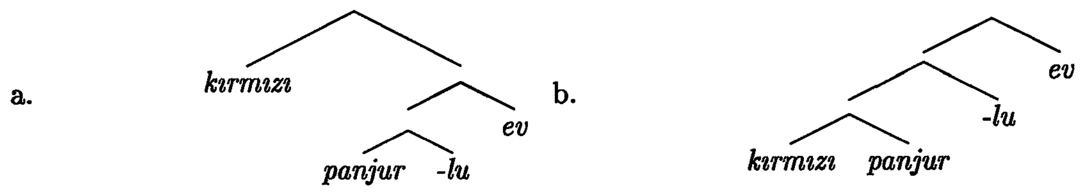


Figure 3.2: Alternative Bracketings of *kırmızı panjur-lu ev*

```
b. kırmızı panjur    -lu      ev
   -------  ----  --------   --
    N/N      N    (N/N)\N    N
   -----------> 
        N
   -------------------------<
             N/N
   ---------------------------------->
                 N
```

'the house with red shutter'

The scope of a morpheme may also be a verb phrase as shown in example (22). Subordinate markers in Turkish are of this category. In the example the scope of *-ma* is the phrase [*bebeğin ağla*].

```
(22) Bebeğ  -in      ağla        -ma-sı                kadın   -ı        yor          -du
     baby   -AGR     cry    -GERUND-POSS.3s           woman   -ACC    fatigue       -TENSE
     ---   ------   ------   -----------------        -----  ------  ----------  ----------------
      N    NP1\N    S\NP1    N\NP1\(S\NP1)             N     NP2\N   S|NP1|NP2   S\NP1\(S\NP1)
     --------------<  ---------------------------<   ------------<
         NP1                    N\NP1                    NP2
     ---------------------------------------------<  ------------------------------<
                        N                                     S\NP1
     ---------------------------------------------T  -------------------------------------------------<
                     S/(S\NP1)                                        S\NP1
     ------------------------------------------------------------------------------------------------<
                                                   S
```

'The crying of the baby exhausted the woman.'

A new lexical operator, which gives more flexibility to CCG, is added to CCG in this model. It is called *directional underspecification* (Bozşahin, 1998), and is needed when arguments of a functor can scramble to either side of the functor. It is for handling the free word order in Turkish. If only backward slash (\) and forward slash (/) were used, then even for a intransitive verb in Turkish there should be two lexical entries in the lexicon. When transitive verbs, ditransitive verbs, and type-raised lexical items are considered, the lexicon becomes very large. However, with the introduction of this directional underspecification, only one entry for e.g. a ditransitive verb is enough.

The neutral slash, |, is the lexical operator used for this purpose. There is no need to add new CCG rules with this operator; it is instantiated to either forward slash (/) or backward slash (\) in derivations. The categories of intransitive, transitive, and ditransitive verbs in Turkish are as follows when neutral slash is used:

(23) a. $IV = S|NP_1$

b. $TV = S|NP_1|NP_2$

c. $DV = S|NP_1|NP_3|NP_2$

Derivation of a Turkish sentence with a ditransitive verb using these categories is shown in example (24). Here, all nouns are type-raised. Their type-raised categories are written as S/IV, TV/DV, etc. for ease of exposition (Bozşahin, 1998).

(24)
```
  Kız        kedi-ye    süt-ü      ver-di
  girl.NOM   cat-DAT    milk-ACC   give-TENSE
  --------   --------   --------   ----------
   S/IV       TV/DV      IV\TV        DV
             ---------------------Bx<
                  IV/DV
             ------------------------------->
                        IV
  -------------------------------------------->
          S: give cat milk girl
```
'The girl gave the milk to the cat.'

Only inflectional morphology is the subject of this study. Since derivational morphology forms new words, it is not included. Therefore, by morphology we mean inflectional morphology from now on.

3.3 The Lexicon

The lexicon used in this study has the same structure proposed in Bozşahin (1999). Since this is a morpheme-based formalism, morphemes have the same representation as words in terms of syntactic, semantic, phonological, and morphological aspects. Two items from the lexicon are shown in (25) in Prolog notation, where one is a free morpheme (that is, a word), and the other is a bound morpheme. '˜' is the juxtaposition operator for the PAS.

(25) a. [gör],

((s,{(Tense,Num,Per,Gender),≤s-base},DAGSyn) : (see˜X˜Y,DAGSem) |

```
(np,{(1,nom,Num1,Per1,Gender1),≤phrase},DAGSyn1) : (Y,DAGSem1)|
(np,{(2,acc,Num2,Per2,Gender2),≤phrase},DAGSyn2) : (X,DAGSem2),
phon(('gör',[])),
morph(('v',(free,concat),[]))
).
```

b.
```
[ler,lar],
((n,{(Index,Case,pl,Per,Gender),≤n-num},DAGSyn) : (plu˜X,DAGSem) \
(n,{(Index1,Case1,sg,Per1,Gender1),≤n-base},DAGSyn):(X,DAGSem1),
phon(('lAr',[])),
morph(('-PLU',(bound,affix),[]))
).
```

Every lexical entry has an ordered three-tuple description, (*Category:Type, Phon, Morph*), along with a list of all its possible allomorphs. For example, the plural morpheme has two possible surface forms as a result of vowel harmony in Turkish. The *Category:Type* pairing includes category and the semantics of the lexical items. This part also holds some other information which will be explained later.

There are three basic categories: *N, NP*, and *S*. All have some specific information. *S*s have quadtuples shown in parenthesis in (25). These are: tense, person, number, and gender.*N*s and *NP*s have quintuples: index, case, number, person, and gender. All of these information may be variables in the lexicon, and can get values via unification. Using both index and case is necessary for Turkish. A transitive verb can take an accusative or a dative *NP* as its object. If only index information is used, then ill-formed derivations will be possible. Only case information is not sufficient, either. Because an NP_{dat} is not necessarily an argument, but $NP_{2,dat}$ is. The latter can be type-raised as $(S\backslash NP_1)/(S\backslash NP_1\backslash \mathrm{NP}_{2,dat})$, but this is not the category of NP_{dat} as an adjunct.

Item-particular knowledge, such as selectional restrictions, are represented as directed acyclic graphs (DAGs), and checked by DAG unification. Syntactic DAG and semantic DAGs are separate items.

Every category in lexical entries are decorated with lattice information shown as the last element in curly braces in (25). This represents a *hypocategory*, which is used to provide a finer level of control. Hypocategories represent the grammatical information state of the linguistic object. They are further refinements of basic categories of the CCG used for the distinction in form-meaning correspondence. In order to ensure category unity, lattice condition is imposed on the hypocategories. Bozşahin (1999) describes this as follows:

> The term 'hypocategory' is suggested by the relationship between a category X, hypocategory κ, and the lattice L: A decoration of X with hypocategory κ (denoted X^{κ}) implies that $\kappa \leq \top$ where $\top$ is the top element of L. In other words, category X without decoration (which is equivalent to $X^{\top}$) subsumes all decorated versions of X.

Using hypocategories results in a natural treatment of morphosyntactic composition without resorting to nonmonotonic operations.

The categories, as a result of inflections and other grammatical markings, undergo stages of specification. For example, although a singular *N* has the same category with a plural *N*, they cannot fulfill the same requirements in any grammatical context. The distinction is provided by hypocategories. The categories of singular and plural nouns are $N^{=n-base}$ and $N^{=n-num}$ respectively.

The CCG rules should be restated when hypocategories are added to the categories. Let us assume that the notation X^{R_κ} specifies the binary relation R on X such that category X has the hypocategory κ subject to lattice relation R on categories (Bozşahin, 1999). Then the forward application rule of CCG becomes as below with hypocategory decorations:

(26) $\dfrac{X^{R_x\kappa_x}/Y^{R_y\kappa_y} \quad Y^{R_{y'}\kappa_{y'}}}{X^{R_x\kappa_x}}>$ if $\kappa_{y'}R_y\kappa_y$ holds in the lattice

There are two binary lattice relations: (=) for strict control and ($\leq$) for flexible control. For instance, the category $N^{<=n-num}$ means that this is a category with hypocategory *n-num* which must category-include x, for some x in the lattice ($x \leq$ *n-num*). The x argument of the binary relation ($\leq$) is filled in during derivations. If the category is $N^{=n-num}$, then this is a category with hypocategory *n-num* which must category-equal x, for some x in the lattice (x=*n-num*). In this case, x must be *n-num*.

Hasse diagram for the category lattice of Turkish is shown in Figure 3.3. Immediate succersors of the bottom ($\perp$) corresponds to unmarked stages of basic categories (*N*, *NP*, *S*).

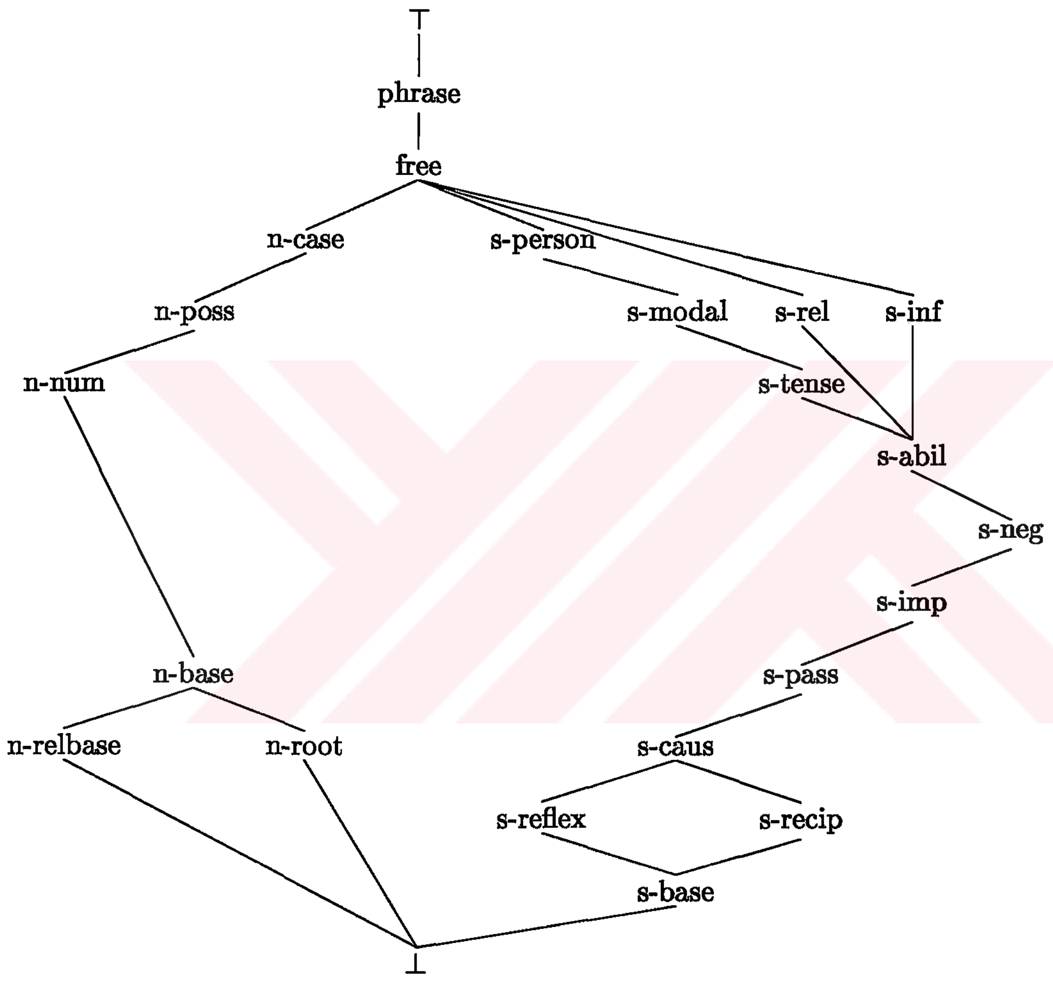


Figure 3.3: The Hasse Diagram for Turkish

Using hypocategories has other advantages as well. For example, in order to model finite and infinite subordination in Turkish, the phenomena that are syntactically similar but semantically different, hypocategories are very useful. Hypocategories may also be used to prevent the permutation problem in endotypic categories, such as $X\backslash X$, of many inflectional morphemes. Moreover, hypocategories simplify the account of recursive nature of some inflectional morphemes in Turkish, like *-ki* relativization.

For example, let us assume that the two optional morphemes m_1 and m_2 both have the category $N\backslash N$ and that they can only appear in that order. Hence, *stem*, *stem* m_1, *stem* m_2, and *stem* m_1 m_2 are words where *stem* m_2 m_1 is not. When the above endotypic categories are decorated with hypocategories, derivation of only correct orders can be achieved without any further operation. With the hypocategories, the categories of the morphemes become:

(27) $m_1 := N^{\kappa_1'}\backslash N^{\kappa_1}$

$m_2 := N^{\kappa_2'}\backslash N^{\kappa_2}$ where $\kappa_i < \kappa_{i'}$ and $\kappa_{1'} \leq \kappa_2$

The illegal derivation is blocked in (28d) and all other derivations are possible (28a-c).

(28) a.

$$\begin{array}{lll} \textit{stem} & -m_1 & -m_2 \\ \overline{N^{\leq s}} & \overline{N^{\kappa_1'}\backslash N^{\kappa_1}} & \overline{N^{\kappa_2'}\backslash N^{\kappa_2}} \\ \multicolumn{2}{c}{\overline{\qquad\qquad\qquad}{<}} & \\ \multicolumn{2}{c}{N^{\leq \kappa_1'}} & \\ \multicolumn{3}{c}{\overline{\qquad\qquad\qquad\qquad\qquad\qquad}{<}} \\ \multicolumn{3}{c}{N^{\leq \kappa_2'}} \end{array}$$

b.

$$\begin{array}{c} \textit{stem}\ -m_1 \\ \overline{\qquad\qquad}{<} \\ N^{\leq \kappa_1'} \end{array}$$

c.

$$\begin{array}{c} \textit{stem}\ -m_2 \\ \overline{\qquad\qquad}{<} \\ N^{\leq \kappa_2'} \end{array}$$

d.

$$\begin{array}{ll} \textit{stem}\ -m_2 & -m_1 \\ \overline{\qquad\qquad}{<} & \\ N^{\leq \kappa_2'} & \\ \multicolumn{2}{c}{\overline{\qquad\ *** \qquad}{<}} \\ \multicolumn{2}{c}{\textit{because}\ \kappa_{2'} \not\leq \kappa_1} \end{array}$$

It is necessary to prevent a misunderstanding about hypocategories: they are not representations of morphotactic rules of the language. The morphotactics are part of the syntax in the categorial framework since they affect the PAS.

Table 3.1: Attachment Calculus for Turkish

		Argument Type	
Process	Functor	free	bound
Affixation	free	-	(concatenation, free)
	bound	(concatenation, free)	-
Concatenation	free	(concatenation, free)	-
	bound	-	-
Clitic	free	-	(concatenation, free)
	bound	(concatenation, free)	-

Since morphological and syntactic composition is realized in the same system, it is necessary to include further refinements of concatenation of two lexical entries. The type of an affix, for example, is indicated by the directionality of the functor. If the category of a bound morpheme is $X\backslash Y$, then it is a suffix, and it is a prefix if its category is X/Y. Property of the morpheme and its attachment characteristics contribute to derivability by an attachment calculus. The attachment calculus for Turkish is shown in Table 3.1 (Bozşahin, 1999). The table states that, for instance, a free and a bound morpheme derives a free morpheme of concatenation as a result of affixation process if their categories match. Also, two free morphemes cannot derive another morpheme under affixation process.

The process and type information is stored in *Morph* part of the lexical entries. Therefore, during derivation *Morph* dimension of the lexical entries are also used.

Phon part contains the phonological form of the entry. Meta-phonemes are indicated by upper-case letters. The meta-phonemes used in the lexicon and their surface form realizations are given in Table 3.2. Parentheses indicate insertion/deletion depending on the previous segment. For example, *A* in the plural morpheme in (25b) indicates that this will be *a* or *e* in the surface form when vowel harmony is applied. The *Phon* part of the accusative case marker is *(y)I.* The optional buffer segment *(y)* indicates that the buffer consonant *y* can be dropped if it is attached to stem ending with a consonant, otherwise the buffer consonant will be used. Since *I* is the meta-phoneme for *ı, i, u, ü,* this entry has eight possible surface forms, hence its list of allomorphs consists of *ı, i, u, ü, yı, yi, yu, yü.* Instead of selecting the appropriate allomorph from

Table 3.2: Meta-Phonemes Used In the Lexicon

Meta-Phoneme	Description
A	back (a) or front (e)
I	high vowels (ı, i, u, ü)
B	voiced (b) or voiceless (p)
C	voiced (c) or voiceless (ç)
D	voiced (d) or voiceless (t)
G	(ğ) or (k)

the allomorph list, *Phon* part is used during printing in generation. The latter method is preferred over the first one because of efficiency reasons. In the parser, using the allomorphs is better again for efficiency reasons.

Morph part shows part of speech of the lexicon and gives attachment information. Whether the morpheme is bound or free is indicated in this part.

As stated in Section 2.2, CCG, like other flexible CGs, suffers from spurious ambiguity problem. In Bozşahin's parser (1998) the solution Eisner (1996) proposed is used to deal with the extra parses of a CCG derivation. Eisner's technique finds exactly one parse in each semantic equivalence class of allowable parses. All other parses are suppressed by simple normal-form constraints that are enforced throughout the parsing process. All CCG rules are rewritten using the labels of the constituent categories Eisner used. The aim of this approach is to block derivations which cause spurious ambiguity.

3.3.1 The Specification of the Lexical Entries

The lexicon in Bozşahin (1999) includes a sampler of nouns, verbs, adjectives, and grammatical morphemes for inflections, subordinate, and relative markers, compounds, etc. During the generation process, the lexical information plays a central role, therefore we feel it necessary to describe the characteristics of the lexical entries in detail. In each section below, the category-PAS assignments in that lexicon are described, full-blown representation of attachment and category control is shown, and derivations with given category assignments are exemplified. Hypocategories are not shown in derivations for ease of exposition. The binding theory in Steedman (1996) is assumed for the PAS. Functional constants in the PAS will be written in capital letters. As we explain in Chapter 4, type-raised lexical entries are not used during the generation process, therefore, type-raised lexical items will not be described in the following sections.

Most of the bound morphemes have phrasal scopes. In a word-based formalism, rebracketing will be necessary to handle this feature. However, in a morphosyntactic model, phrasal scope is not a problem.

3.3.1.1 Nouns and Pronouns

Nouns and pronouns are not functions; their PASs do not have arguments. The only difference between a noun and a pronoun lexical entry is in *Morph* dimension. Nouns have *n* and pronouns have *pn* in their *Morph* part. This distinction is useful in subject and object dropping; only pronouns can drop. Otherwise nouns and pronouns are identical. There is no distinction between proper nouns and common nouns in the representation. Nouns and pronouns are free morphemes. The category assignment for nouns and pronouns is:

(29) (n or pn) := $< N^{\leq n-root} : x$, (free, concatenation)>

3.3.1.2 Adjectives

Adjectives are functions over a noun or a noun group. They must precede the nouns they are modifying. Adjectives are free morphemes. Therefore, the full blown representation for adjectives is:

(30) adj := $< N^{\leq n-phrase} : fx \,/ N^{\leq free} : x$, (free, concatenation)>

A noun or a noun group can have more than one adjective that modifies it. Below is a derivation of a phrase with two adjectives and a noun.

(31)
$$
\begin{array}{ccc}
\textit{küçük} & \textit{mavi} & \textit{ev}\\
\text{small} & \text{blue} & \text{house}\\
\overline{N/N} & \overline{N/N} & \overline{N}\\
 & \multicolumn{2}{c}{\overline{}>}\\
 & \multicolumn{2}{c}{N}\\
\multicolumn{3}{c}{\overline{}>}\\
\multicolumn{3}{c}{N}
\end{array}
$$

'small blue house'

3.3.1.3 Plural Marker

The plural marker in Turkish is the suffix *-lAr*. It is affixed to the noun or the noun group and yields a nominal stem. The only consideration in pluralization is that, the argument of the plural marker should be singular, and the resulting stem should be plural. Lexical category assignment for the plural marker is:

(32) -PLU := < $N_{pl}^{\leq n-num}$: PLU x $\backslash N_{sg}^{\leq n-base}$: x, (bound, affixation)>

In (33c), the scope of the plural marker is phrasal. (33b-c) show two possible derivations for the ambiguous phrase. In (33b), the derivation gives the semantics 'the set of room sets which are blue', whereas in (33c) we get the meaning 'set of individual blue rooms', which is the correct semantics.

(33) a.

$$\begin{array}{cc} \textit{oda} & \textit{-lar} \\ \text{room} & \text{-PLU} \\ \hline N & N\backslash N \\ \hline \multicolumn{2}{l}{N : PLU\ room \quad <} \end{array}$$

'rooms'

b.

$$\begin{array}{ccc} \textit{mavi} & \textit{oda} & \textit{-lar} \\ \text{blue} & \text{room} & \text{-PLU} \\ \hline N/N & N & N\backslash N \\ \cline{2-3} & \multicolumn{2}{c}{N \quad <} \\ \hline \multicolumn{3}{l}{N : blue\ (PLU\ room) \quad >} \end{array}$$

'blue rooms'

c.

$$\begin{array}{ccc} \textit{mavi} & \textit{oda} & \textit{-lar} \\ \hline N/N & N & N\backslash N \\ \cline{1-2} \multicolumn{2}{c}{N \quad >} & \\ \hline \multicolumn{3}{l}{N : PLU\ (blue\ room) \quad <} \end{array}$$

'blue rooms'

3.3.1.4 Case Markers

Case marking in Turkish indicates grammatical functions. Case marking is realized by bound case morphemes. When a constituent is case marked, its function in the sentence is determined, and its position is not fixed. Therefore, the grammatical function of an

Table 3.3: Turkish Case Markers

Case	Genotype	Index
Nominative	nom	1
Genitive	gen	1
Accusative	acc	2
Dative	dat	2,3
Ablative	abl	4
Locative	loc	6
Comitative	com	7

NP is identified by case markers. The syntactic category of the *NP* can be specified using genotype indices once its grammatical function is known. The case markers in Turkish and their genotypes and indices are given in Table 3.3.

Third person possessive *NP*s and the other *NP*s take different morphemes as case markers (34).

(34)	a.	*oda-yı*	b.	*ben-im*	*oda-m-ı*	c.	*kız-ın*	*oda-sı-nı*
		room-ACC		I-GEN.1	room-POSS.1s-ACC		girl-GEN.3	room-POSS.3s-ACC2
	d.	*oda-da*	e.	*ben-im*	*oda-m-da*	f.	*kız-ın*	*oda-sı-nda*
		room-LOC		I-GEN.1	room-POSS.1s-LOC		girl-GEN.3	room-POSS.3s-LOC2
	g.	**oda-nı*	h.	**kız-ın*	*oda-sı-yı*			
		room-ACC2		girl-GEN.3	room-POSS.3s-ACC			

This distinction is specified in the lattice condition. Two morphemes are defined for each case marker and these two morphemes are symbolized as -CASE and -CASE2. The only difference between these two is in the degree of control. The case markers for third person possessive *NP*s require strict control, whereas other *NP*s require flexible control. By assigning the hypocategory $N^{=n-poss}$ to -CASE2, the derivation of (34g-h) are blocked.

(35) -CASE := < $NP^{\leq phrase}\!: x \ \backslash N^{\leq n-num}\!: x$, (bound, affixation)>

-CASE2 := < $NP^{\leq phrase}\!: x \ \backslash N^{=n-poss}\!: x$, (bound, affixation)>

Case affixation is not a word-based operation; the scope of a case marker may be a noun group. This is most evident when an entire case-marked subordinate clause is an argument of the verb (36). Accusative *-nü* delimits this nominalized clause. The semantics of the sentence also indicates a morpheme-based derivation.

(36) *Kadın* [*çocuğ-un* *gül-düğü*]*-nü* *söyle-di*
woman child-GEN3 laugh-SUB2g -ACC2 say-TENSE
'The woman said that the child laughed.' = *say(laugh child)woman*

3.3.1.5 Genitive Constructions

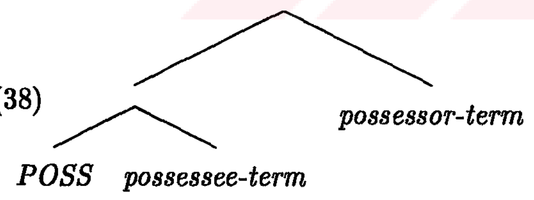
The genitive marks dependencies between a possessor and a possessee. The genitive case marks the possessor and the possessive marker marks the possessee.

(37) a. *çocuğ-un*$_i$ *kendi-si*$_i$
child-GEN.3 self-POSS.3s
'the child himself'

b. **kendi*$_i$ *çocuğ-u*$_j$

c. **kendi-nin*$_i$ *çocuğ-u*$_i$

The genitive-possessive construction in Turkish has the structure in (38) as a result of binding asymmetries between the specifier and the head (37). The possessor scopes over the possessee as can be seen from the binding constraints.

(38)


The possessor and the possessee must agree on person and number (40a). When a sequence of genitive nouns is used in specifier position (nested genitives), each genitive-marked noun requires a possessive marked noun to follow (40b). The possessive marker is also a functor as genitive marker since it enforces agreement with the specifier. Its scope is the possessee. Both genitive and possessive markers are bound morphemes yielding nominal stems via affixation. The scope of these markers may be a noun or a noun group which is determined by derivations. In example (40a), the genitive marker ranges over the noun group *büyük hanım* and the possessive marker ranges over the noun group *takma diş*.

The category of the stem that the genitive marker is attached to and the result of the genitive inflection must category-include *n-poss* ($\leq$*n-poss*). As for the category of the possessee argument, it must be (=*n-poss*) since it must be inflected with the possessive marker. *Adam-ın çocuk* or *adam-ın çocuk-lar* are not grammatical in Turkish.

The category assignment of genitive and possessive markers are as below. *p* and *n* represent person and number.

(39) $\text{-GEN}_{pn} := < N^{\leq n-poss} : \text{POSS } xy \,/(N_{pn}^{=n-poss} : \text{POSS } xy \setminus N_{pn}^{=n-poss} : x) \setminus N_{\overline{pn}}^{\leq n-poss} : y,$ (bound, affixation)>

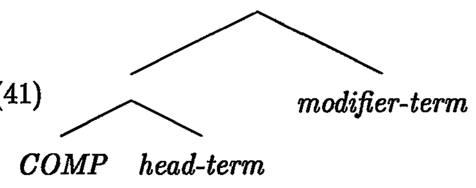
$\text{-POSS}_{pn} := < N_{pn}^{=n-poss} : \text{POSS } xy \setminus N_{pn}^{=n-poss} : x \setminus N_{\overline{pn}}^{\leq n-num} : x,$ (bound, affixation)>

(40) a.
```
büyük hanım     -ın        takma     diş      -i
 old   lady   -GEN.3    artificial tooth -POSS.3s
 ---   ----   ---------  ---------  -----  --------
 N/N    N    N/(N\N)\N     N/N       N     N\N\N
 ------------>            ---------------->
       N                         N
 ----------------------<  ------------------------<
        N/(N\N)                   N\N
 ------------------------------------------------->
        N : POSS(artificial tooth)(old lady)
              'the old lady's artificial tooth'
```

b.
```
ben     -im        ev     -im       -in       bahçe     -si
 I     GEN.1     house -POSS.1s    GEN.3     garden   POSS.3s
 -   ---------   ---  --------- ---------   ------  --------
 N   N/(N\N)\N    N     N\N\N    N/(N\N)\N    N      N\N\N
 --------------<  ---------------< ------------------<
    N/(N\N)             N\N              N\N
 --------------------------------->
                N
 ----------------------------------------------------<
                 N/(N\N)
 ------------------------------------------------------------------>
            N : POSS garden (POSS house me)
                        'my house's garden'
```

3.3.1.6 Syntactic Compounds

Syntactically, genitive constructions and syntactic compounds are similar, however semantically they are different. Therefore, they should have separate representations. The main difference between genitives and compounds is the nonreferentiality and nonspecificality of the head and the modifier in the PAS of the compounds.

(41)


In syntactic compounds, the head takes the compound marker which is a bound morpheme. The head cannot be inflected on any grammatical functions and it cannot be a phrase (42b). The modifier does not take any suffix and it can be noun group (42c). The entire compound can only take case-markers (42d-f), but the case markers that are applied to the possessive *NP*s, that is -CASE2 morphemes, can be affixed to the compounds (42d-e).

(42) a. *kapı kol-u*
door handle-COMP
'door handle'

b. **kapı altın kol-u*
door golden handle-COMP

c. *altın kapı kol-u*
golden door handle-COMP

d. *kapı kol-u-nu*
door handle-COMP-ACC2

e. **kapı kol-u-yu*
door handle-COMP-ACC

f. **kapı kol-u-su*
door handle-COMP-POSS.3s

In the genitive constructions involving compounds, compound markers have two functions: compounding and agreement. These constructions cannot take a possessive marker (43) (Bozşahin, 1999).

(43) a. *banka-nın faiz oran-ı*
bank-GEN.3 interest rate-COMP.POSS
'interest rate of the bank'

b. **banka-nın faiz oran-ı-sı*
bank-GEN.3 interest rate-COMP-POSS.3s

Therefore, there should be two category-type assignments for compound markers:

(44) -COMP $:= < N^{=n-poss} : \text{COMP}\ xy \backslash N^{\leq phrase}\!: y \backslash N^{=n-root}\!: x,$ (bound, affixation)>

-COMP2 $:= < N^{=n-poss} : \text{POSS(COMP}\ xy)z \backslash N^{=n-poss}\!: z \backslash N^{\leq phrase}\!: y \backslash N^{=n-root}\!: x,$ (bound, affixation)>

The plural compounds are composite functions as explained in Section 3.2. Thus, the morpheme *-lArI* has the following lexical categories:

(45) -COMP $:= < N^{=n-poss} : \text{PLU COMP } xy \, \backslash N^{\leq phrase}: y \, \backslash N^{=n-root}: x,$ (bound, affixation)>

-COMP2 $:= < N^{=n-poss} : \text{POSS PLU (COMP } xy)z \, \backslash N^{=n-poss}: z \, \backslash N^{\leq phrase}: y \, \backslash N^{=n-root}: x,$ (bound, affixation)>

3.3.1.7 Verbs

Since Turkish is a free constituent order language, all permutations of the predicate and its arguments in a sentence are grammatical, subject to constraints on contextual and discourse relations (e.g. definiteness of arguments). Grammatical function mapping of the arguments are done using case marking. Nominal and genitive cases marks subjects and accusative and dative cases mark objects in Turkish.

A transitive verb can take an accusative or a dative marked object. In ditransitive verbs, NP_2 is accusative case marked, and NP_3 is dative case marked. The case-grammatical function correspondence is made explicit in the lexical entries. For instance, if a verb subcategorizes for a dative object, its category is $S|NP_1|NP_2$ rather than $S| \, NP_1|NP_3$ which the dative object seems to suggest. Therefore, all transitive verbs have the category $S| \, NP_1 \, |NP_2$; the case for object is provided for each lexical entry individually.

The category assignment for verbs are:

(46) intransitive verb $:= < S^{\leq s-base} : fx \, |NP^{\leq phrase}_{1,nom}: x,$ (free, concatenation)>

transitive verb $:= < S^{\leq s-base} : fyx \, |NP^{\leq phrase}_{1,nom}: x \, |NP^{\leq phrase}_{2}: y,$ (free, concatenation)>

ditransitive verb $:= < S^{\leq s-base} : fzyx \, |NP^{\leq phrase}_{1,nom}: x \, |NP^{\leq phrase}_{3,dat}: z \, | \, NP^{\leq phrase}_{2,acc}: y,$ (free, concatenation)>

3.3.1.8 Relativization

There are two strategies for forming relative clauses: subject participle strategy (SP) and object participle strategy (OP). In this formulation, there is no empty categories, traces, or movement in the representation of relatives in the PAS. A functional symbol

ANA, which equates the PAS of the relativized noun anaphorically, is used instead of variables.

In Turkish, the noun on which the relativization is performed is placed after the relative clause, that is, relativization is head-final. Relativization morphemes are bound morphemes. Since the scope of any further grammatical marking on the head noun is the entire relative clause, the head noun should have strict constraint (*=n-base*) for category control over the lattice. Relativization in Turkish is morpheme-based semantically; the relativization morpheme range over the entire relative clause. In (48a), rebracketing of the relative clause would be required in a word-based formalism in which the morpheme *-an* will be attached to *bak*. Subject's specifier can also be relativized.

Although the agreement marker in OP strategy is a functor over the relativization morpheme, the relativization morpheme triggers the agreement in the OP strategy. Therefore, relativization and agreement morphemes can be treated as a composite affix (-REL.OP). The full blown representation of the relativization morphemes are:

(47) -REL.SP := $< N^{\leq n-base}$: REL$(f(\text{ANA } x))x \; / N^{n-base}\!: x \; \backslash (S\!: fx | NP_1^{\leq phrase}\!: x)$, (bound, affixation)>

-REL.SP := $< N^{\leq n-base}$: REL$(f(\text{POSS}x(\text{ANA } y)))y \; / N^{\leq n-base}\!: y) \backslash$ $(N^{=n-poss}$: POSS $xz \; \backslash N^{=n-poss}\!: x) \; \backslash (S\!: f(xy) \; | NP_1^{\leq phrase}\!: xy)$, (bound, affixation)>

-REL.OP := $< N^{\leq n-base}$: REL$(f(\text{ANA } x)y)x$ /fs $N^{\leq n-base}\!: x) \backslash NP_{1,gen,pn}^{\leq phrase}\!: y \backslash$ $(S\!: fxy \; | NP_1^{\leq phrase}\!: y \; | NP_2^{\leq phrase}\!: x)$, (bound, affixation)>

-REL.OP := $< N^{\leq n-base}$: REL$(in(fy)(\text{ANA } x))x$ /fs $N^{\leq n-base}\!: x) \backslash NP_{1,gen,pn}^{\leq phrase}\!: y \backslash$ $(S\!: fy \; | NP_1^{\leq phrase}\!: y)$, (bound, affixation)>

Below are derivations for subject (48a), object (48b), and adjunct (48c) relativizations with the defined category assignments. In (48d), the relativization of subject's specifier is shown.

(48) a.

araba	*-ya*	*bak*	*-an*	*çocuk*	*-lar*
car	-DAT	look	-REL.SP	child	-PLU
N	$NP_{2,dat}\backslash N$	$S\vert NP_1\vert NP_{2,dat}$	$N/N\backslash(S\vert NP_1)$	N	$N\backslash N$

$NP_{2,dat}$ <

$S|NP_1$ <

N/N <

N >

$N : PLU(REL(look\ car(ANA\ child))child)$ >

'the children that looks at the car'

b.

çocuğ	*-un*	*vur*	*-duğu*	*top*
child	-GEN.3	kick	-REL.OP	ball
N	$NP_{1,gen}\backslash N$	$S\vert NP_1\vert NP_2$	$N/N\backslash NP_{1,gen}\backslash S\vert NP_1\vert NP_2$	N

$NP_{1,gen}$ < $N/N\backslash NP_{1,gen}$ <

$n>N$ <

$N : REL(kick(ANA\ ball)child)ball$ >

'The ball that the child kicked'

c.

ahmet	*-in*	*uyu*	*-duğu*	*oda*
ahmet	-GEN.3	sleep	-REL.OP	room
N	$NP_{1,gen}\backslash N$	$S\vert NP_1$	$N/N\backslash NP_{1,gen}\backslash S\vert NP_1$	N

$NP_{1,gen}$ < $N/N\backslash NP_{1,gen}$ <

$n>N$ <

$N : REL(in(sleep\ ahmet)(ANA\ room))room$ >

'The room that Ahmet slept in.'

d.

kedi	*-si*	*yüz*	*-en*	*kız*
cat	-POSS.3s	swim	-REL.SP	girl
N	$N\backslash N\backslash N$	$S\vert NP_1$	$(N/N)\backslash(N\backslash N)\backslash(S\vert NP_1)$	N

$N\backslash N$ < $(N/N)\backslash(N\backslash N)$ <

N/N <

$N : REL(swim(POSS\ cat(ANA\ girl)))girl$ >

'The girl whose cat swims'

3.3.1.9 -ki Relativization

As we mentioned in Section 2.1, *-ki* relative suffix, which is a bound morpheme, can recursively generate indefinitely long words in Turkish. *-ki* relative suffix must be attached to a locative or genitive noun. It has pronominal use for both locative (49a) and genitive (49b) nouns and anaphoric (49c) use for the locative nouns (Bozşahin, 1999).

```
(49) a. bahçe    -de          -ki
        garden   -LOC         -REL
        ------  -------    -----------
          N     NPloc\N     N\NPloc
        ---------------<
            NPloc
        ------------------------------<
        N : REL(at(PRO garden)PRO
        'The one that is at the garden'

     b. bahçe    -de        -ki        kedi
        garden   -LOC       -REL       cat
        ------  -------  ----------   -----
          N     NPloc\N  N/N\NPloc      N
        ---------------<
            NPloc
        --------------------------<
                 N/N
        ----------------------------------->
        N : REL(at(ANA cat)garden))cat
        'The cat that is at the garden'

     c. bahçe    -nin            -ki
        garden   -GEN.3          -REL
        ------  ---------   --------------
          N     N/(N\N)\N   N\(N/(N\N))
        -------------------<
             N/(N\N)
        ------------------------------------<
         N : REL(POSS PRO garden)PRO
        'The one that belongs to the garden'
```

Although a nominal stem with *-ki* suffix can undergo all nominal inflections like a nominal root, there is one difference: the stem with *-ki* suffix should receive -CASE2 suffix not -CASE suffix (50).

(50) a. *gemi-de-ki-ni* b. *gemi-yi*

shiP-LOC-NREL-ACC2 ship-ACC

c. **gemi-de-ki-yi* d. **gemi-ni*

ship-LOC-NREL-ACC ship-ACC2

This problem is solved using the hypocategories. Assigning the lattice (*=n-relbase*) to the result of *-ki* relativization eliminates (50c) and (50d). *-ki* suffix ranges over the entire case-marked noun which is a phrase. Again, this suggests a morphosyntactic formalism.

The category assignments for pronominal and anaphoric *-ki* relativization with these are given in (51).

(51) -PROki := < $N^{=n-relbase}$: REL(atPRO x)PRO $\backslash NP_{loc}^{\leq phrase}$: $at\ x$, (bound,affixation)>

-ANAki := <($N^{=n-relbase}$: REL(at(ANA x)y)x /$N^{\leq n-num}$: x)$\backslash NP_{loc}^{\leq phrase}$: $at\ y$, (bound, affixation)>

-PROki := < $N^{=n-relbase}$: REL(POSS PRO x)PRO $\backslash N_{gen}^{\leq n-poss}$: POSS yx / ($N_{gen}^{=n-poss}$: POSS zx $\backslash N_{gen}^{=n-poss}$: w), (bound, affixation)>

3.3.1.10 Subordination

In Turkish, subordinate markers are bound morphemes that may have the scope over a phrase. Subordinate clauses are nominalizations, hence they carry grammatical functions. The resulting subordinate clause behaves as a noun, but can take only case inflections. The scope of these morphemes are the entire subordinate clause. In a word-based approach, this may be achieved via rebracketing.

Subordinate clauses, which are strictly verb-final, can form via gerundive or infinitive suffixes. Gerundive forms require genitive marking on the embedded subject and appropriate possessive marking on the subordinate verb. The arguments for gerundive or infinitive type of subordination functors are different.

The category control over the lattice is (≤*n-poss*) since the resulting nominal stem of subordination can only take case suffix. The subordinate verb cannot be finite, so its category is (≤*s-abil*). In the lexicon, the subordination and the agreement on the subordinate verb are composed. The reason for this affix composition is to ensure the verb-final characteristic of subordination. Hence, the category-assignment for subordinate morphemes are:

(52) -SUB1i := < $N^{\leq n-base}$: fx $\backslash$($S^{\leq s-abil}$: fx $\backslash NP_{1}^{\leq phrase}$: x, (bound, affixation)>

-SUB1g:= < $N^{\leq n-poss}$: fx $\backslash NP_{1,gen}^{\leq phrase}$: x $\backslash$($S^{\leq s-abil}$: fx $|NP_{1}^{\leq phrase}$: x, (bound, affixation)>

-SUB2i := < $N_{i\neq 1}^{\leq n-base}$: fx $\backslash$($S^{\leq s-abil}$: fx $\backslash NP_{1}^{\leq phrase}$: x, (bound, affixation)>

-SUB2g:= < $N^{\leq n-base}_{i\neq 1}\!: fx \backslash NP^{\leq phrase}_{1,gen} \backslash (S^{\leq s-abil}\!: fx \,|NP^{\leq phrase}_{1}\!: x$,

(bound, affixation)>

3.3.1.11 Adjuncts

Adjuncts have the endotypic category $S|S$. For instance, the sentence in (54) has a intransitive verb, however there are two *NP*s in the sentence. The first *NP* represents an adjunct. Adjunct cases are locative, ablative, instrumental, and dative. These correspond to *at, from, with*, etc. in English.

(53) -CASE3 := < $S\!: relxy|S\!: x\backslash N^{\leq n-num}\!: y$, (bound, affixation)>

(54)
$$\begin{array}{cccc}
\textit{Araba} & \textit{-da} & \textit{Ahmet} & \textit{uyu-du} \\
\text{car} & \text{-LOC} & \text{Ahmet.NOM} & \text{sleep-TENSE} \\
\overline{N} & \overline{S|S\backslash N} & \overline{S/(S|NP_1)} & \overline{S|NP_1} \\
\multicolumn{2}{c}{\overline{\qquad S|S \qquad}{<}} & \multicolumn{2}{c}{\overline{\qquad\qquad S \qquad\qquad}{>}} \\
\multicolumn{4}{c}{\overline{\qquad\qquad S : at(sleep\ ahmet)car \qquad\qquad}{fa}}
\end{array}$$

'Ahmet slept in the car.'

3.3.1.12 Tense and Person

Tense and person suffixes are bound morphemes that are attached to verbs. They scope over the VP, that is, $S|NP_1$, with the category $VP\backslash VP$ (55). Singular third person has no surface realization in Turkish, so there is no lexical item for it.

(55) -TENSE := <$(S^{\leq s-tense}_{tense}\!: xy|NP^{\leq phrase}_{1}\!: y)\backslash(S^{\leq s-abil}\!: xy|NP^{\leq phrase}_{1}\!: y)$,

(bound, affixation)>

-PERSON := <$(S^{\leq s-person}_{per,num}\!: xy|NP^{\leq phrase}_{1}\!: y)\backslash(S^{\leq s-modal}\!: xy|NP^{\leq phrase}_{1}\!: y)$,

(bound, affixation)>

(56)

Ben	*kitab*	*-ı*	*oku*	*-du*	*-m*
I	book	-ACC	read	-TENSE	-PERS.1s
$S/(S\|NP_1)$	N	$NP_2\backslash N$	$S\|NP_1\|NP_2$	$S\|NP_1\backslash(S\|NP_1)$	$S\|NP_1\backslash(S\|NP_1)$

$$\overline{\qquad\qquad}^{<}\ NP_2$$

$$\overline{\qquad\qquad\qquad\qquad}^{<}\ S|NP_1$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad}^{<}\ S|NP_1$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{<}\ S|NP_1$$

$$\overline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}^{>}\ S : read\ book\ i$$

'I read the book.'

CHAPTER 4

GENERATION OF TURKISH MORPHOSYNTACTIC STRUCTURE

The generation algorithm used in this study is an adaptation of the semantic head-driven generation algorithm (Calder, Reape, and Zeevat, 1989; Shieber et al., 1989; Shieber et al., 1990; van Noord, 1990). This algorithm combines aspects of both top-down and bottom-up generation. It takes advantage of top-down input provided by the user as well as the bottom-up lexical information. The implementation of the generator is done in SICStus Prolog. Although the algorithm is adapted from the semantic-head driven generation algorithm, we do not use the term 'head' in our study, instead we use the term 'anchor'. This is due to the different notion of head in CG (Bouma, 1988). In a context-free rule *VP* $\rightarrow$ *V NP*, *V* (verb) is the head. In CG terms, this is not a rule but application of the verb's category *VP/NP* (=*V*) to *NP*, in which *VP* is the head. But in *VP* $\rightarrow$ *VP PP* where *PP* is considered as adjunct, *VP* is considered the head, which translates in CG to the application of *VP/VP* to *PP*, in which *PP* is the head, because *VP/VP* is endotypic. In CG, there are no type-specific rules, hence no specific grammar rule from which the head can be located; they are anchored in the lexicon, and derived by a syntactic type-free grammar such as $X/Y\ Y \Rightarrow X$.

4.1 Input

The generator gets as input the basic category and the semantic representation of the desired output. It produces as output all possible surface forms of the semantic interpretation and the category. The possible basic categories are *N*, *NP*, *S*, and any partial construction derived from these categories such as *S*/*NP*. The semantic representations are in the form of PAS. The case of the *NP*s, and the tense and person information of *S*s are optional inputs. Case, tense, and person information are not represented in the PAS, they are given in the syntactic features of the lexical entries. Hence, this information should be given in the input in another way. If there is no tense and person information, the output sentence(s) are infinitive, otherwise the appropriate tense marker is attached to the verb. Since only finite verbs can take person suffix, person information should be given along with tense information; it cannot occur alone. If input basic category is an *NP*, case information should be in the input as well, and if the input basic category is an *N*, it cannot be case-marked, so case information cannot be given in input. Finally, whether the subject and/or object of the output surface form will be dropped is declared in input as an optional parameter. This information is valid if the basic category in the input is *S*. Example below shows the input and the output for the subject dropped realization of *give mehmet (POSS book man) she.*

(57) *s-past-dropsub : give mehmet (POSS book man) she*

Mehmed-e	adam-ın	kitab-ı-nı	ver-di
mehmet-DAT	man-GEN.3	book-POSS.3s.ACC2	give-TENSE

'(She) gave the man's book to Mehmet.'

Object drop is shown in the following example.

(58) *s-past-dropobj : put car she (PLU child)*

Çocuk-lar	araba-ya	koy-du
Child-PLU.NOM	car-DAT	put-TENSE

'The children put (it) to the car.'

4.2 Overview of the Generation Algorithm

The generation algorithm works as follows. First, it finds the *anchor* of the output surface form in the lexicon using the basic category and the PAS in the input. The lexical entry whose basic category and PAS unify with the basic category and the PAS in the given input is the anchor. This part of the process is top-down. Then, the arguments of the matched lexical functor (if it is functor) is generated in a bottom-up fashion. The categorial operators reveal the surface ordering of the functor and the arguments. For instance, in $X\backslash Y$, Y preceeds X, whereas in X/Y, X preceeds Y in the surface form. The function that generates the arguments is called recursively until it has found all of the arguments of the anchor. This anchor-driven generation algorithm uses syntactic, semantic, and morphological information in the lexicon.

Let us assume that the input to the generator is $s : p\ a\ b$, and the lexical entry of the anchor has the following category:

$S : p\ x\ y \backslash NP_{1,nom} : y \backslash NP_{2,acc} : x$

After the anchor is found and put in the list of surface forms, the generator attempts to generate all its arguments. Each category-PAS pairing separated by categorial operators in the lexical entry of the anchor is called recursively by the generator. First, the category-PAS pairing after the last (right-most) categorial operator is taken. In the example above, this is ($NP_{2,acc}$: x). Generation algorithm is called recursively with this input with the substitution x/a due to the match of the input with the anchor. The generated string is added to the surface form and this time the generation algorithm is called with ($NP_{1,nom}$: y) with the substitution y/b. After the result is again added to the surface form, only (s: $p\ a\ b$) remains which has already been generated and put in the surface form as anchor. Hence, generation ends. During the generation of the arguments, the same procedure applies: if the argument is a functor, the generation algorithm is called recursively as long as there are arguments to generate.

In the lexical search, the *Category:PAS* information of the lexical entries is fully used. The input to the generator must unify with a lexical entry in all fields. The lattice-consistency is checked for hypocategories, and unification of Directed Acyclic Graphs (DAGs) is required for syntactic and semantic DAGs in the category of the input and the lexical entry. The consistency of the categorial operators is also checked; for instance while searching a category with a neutral slash, the operator in the goal lexical entry can be backward, forward, or neutral slash, while searching a category with a backward slash, the operator in the goal lexical entry can be a backward slash or a neutral slash.

An explanation is needed as to how the categorial operators are used during generation. Categorial operators determine the order of morphemes with respect to each other. If the operator is backward slash (\), the argument is put to the left of its functor; if the operator is forward slash (/), the argument is put to the right of its functor;

and finally if the categorial operator is neutral slash (|), the argument is put first to the left, then to the right of its functor. Hence, for each neutral slash two surface forms are generated.

Tense and person information is not represented in the PAS, but given separately in the input. After the verb is generated, if there is tense and person information in the input, the appropriate tense and person morphemes are found and added to the surface form. In that case, subject-verb agreement is also checked when subject ($NP_{1,nom}$) is generated. The generation process fails if the subject-verb agreement is not fulfilled; that is if input person information and the person information in the lexical entry found as subject conflict. If the input category is *NP*, its case should be given in the input as well. The appropriate case morpheme is found in the lexicon, and generation process goes on as described above.

Turkish nominal case marker has no surface realization, therefore it does not have a lexical entry in the lexicon. In order to generate a nominative-marked noun phrase, we do what we can call *'syntactic type-lowering'* (59), that is we look for an *N* with the same PAS and lexical information as $NP_{1,nom}$.

(59) $NP_{1,nom} : x \;=>\; N : x$

Similarly, direct objects in Turkish can appear without any case marking. This feature of Turkish is also reflected in morphosyntactic generator. This is also achieved by type-lowering. If $NP_{2,acc}$ is to be generated, along with the case-marked *NP*, an *N* with the same PAS and lexical information is looked for in the lexicon. Hence, two surface forms are generated for the same word order if there is accusative case-marked direct object.

Only pronouns can be dropped. If the input has the dropping feature, and the will-be-dropped argument is a pronoun, then this argument is dropped, and is not put in the output surface form.

If the input category is a partial construct, the PAS of the argument that will not be generated should be a variable.

The phonological form of the lexical entries is stored in the *Phon* part. These phonological forms are transformed to surface forms in generation. Meta-phonemes are replaced with their appropriate forms according to morphophonemic rules. During this process, the attachment information (e.g., affixation, cliticization, or concatenation) in the *Morph* part helps to determine the correct form of the meta-phonemes.

According to the vowel harmony rules, the meta-phoneme A in a suffix becomes an *a* if the last vowel in the stem the suffix is attached to is a back vowel, and becomes an *e* if the last vowel in the stem the suffix is attached to is a front vowel. The meta-phoneme I is realized as an *ı* if the last vowel of the stem is a back-unrounded vowel, an as an *i* if the last vowel in the stem a is front-unrounded vowel. I becomes a *u* after back-rounded vowel, and finally becomes a *ü* after a front-rounded vowel.

The phonemes that may drop are written in parentheses in the lexical form to denote that they are buffer phonemes. If the last phoneme of a stem is a vowel, and the stem takes a suffix beginning with an I, this buffer I is dropped. Otherwise, I is realized as one of its forms. Similarly, if the last phoneme of a stem is a consonant, and the stem takes a suffix with a consonant buffer, this buffer phoneme is dropped, otherwise it is not.

When the meta-phonemes B, C, D, G are the last phonemes of the stem, they are resolved as *p, ç, t, k* if they are in stem-final position, or if a suffix beginning with a consonant that does not drop is affixed to the stem. Otherwise, these meta-phonemes are realized as *b, c, d, ğ*. The meta-phoneme D at the beginning of a suffix becomes a *t* if the stem the suffix is attached to ends with a voiceless consonant, or *h*. Otherwise, D is resolved as a *d*.

4.3 Example

In this section, we show the generation of surface form with an example. The operator '˜' is used in PAS as previous, and the operator '-' is used to separate optional input, such as tense or person. The words beginning with a capital letter are the variables as in Prolog, they do not have initial values.

Let us assume that the input to the generator is as follows for *topa vuran çocuk uyudu*:

(60) `s-past-3-sg:sleep˜(REL˜(kick˜ball˜(ANA˜child))˜child)`

The generator first tries to find the anchor whose category is S and whose PAS is unifiable with the one above. The anchor is the verb *uyu* 'sleep', and its lexical representation is:

```
(61) [uyu],
     ((s,{(Tense,Num,Per,Gender),≤s-base},[]):
         (sleep˜((POSS˜(REL˜(kick˜ball˜(ANA˜child))˜child)˜mehmet)),[]) |
     (np,{(1,nom,Num1,Per1,Gender1),≤phrase},[]):
         (REL˜(kick˜ball˜(ANA˜child))˜child,[])
     phon(('uyu',[])),
     morph(('v',(free,concat),[]))
     )
```

Since there is tense information in the input and this is the verb of the sentence, the appropriate tense morpheme, past tense morpheme, is found and added to the verb. The appropriate tense morphemes are searched with their category, which begins with *S*, and their tense information in their category. The past tense morpheme is:

```
(62) [dı, di, du, dü, tı, ti, tu, tü],
     ((s,{(past,Num,Per,Gender),≤s-tense},[]): (Pre˜Arg,[]) |
     (np,{(1,Case,Num1,Per1,Gender1),≤phrase},[]): (Arg,[]) \
     ((s,{(Tense2,Num2,Per2,Gender2),≤s-abil},[]): (Pre˜Arg,[]) |
     (np,{(1,Case3,Num3,Per3,Gender3),≤phrase},[]): (Arg,[])) phon(('DI',[])),
     morph(('-TENSE',(bound,affix),[]))
     )
```

The same procedure is performed for the person morpheme search as well. However, since the third person singular morpheme has no surface realization, nothing is done about the person information in this input.

After this, the generation of the argument(s) begins. The intransitive verb *uyu* has only one argument: its subject. The generation algorithm is called using this argument. The input is the part after the categorial operator. Since this is a $NP_{1,nom}$, type-lowering takes place. The generator will look for an N instead of an $NP_{1,nom}$. Therefore the input is:

```
(n,{(Index1,Case1,Num1,Per1,Gender1),≤phrase},[]):
(REL˜(kick˜ball˜(ANA˜child))˜child,[])
```

The generator finds the SP relativizer morpheme *-(y)An* (63) whose result category unifies with the input category. The person and number information in the input and the person and number fields of this morpheme unify, therefore subject-verb agreement is ensured in the sentence.

(63) [en,an,yen,yan],

```
((n,{(Index,Case,Num,3,Gender),≤n-base},[]):(REL˜(kick˜ball˜(ANA˜child))˜child,[]) /
(n,{(Index1,Case1,Num1,3,Gender1),≤n-base},[]):(child,[]) \
((s,{(Tense,Num2,Per2,Gender2),HypoType},[]):(kick˜ball˜child,[]) |
(np,{(1,nom,Num3,Per3,Gender3),≤phrase},[]):(child,[])),
phon(('(y)An',[])),
morph(('-REL.SP',(bound,affix),[]))
)
```

The category of SP morpheme is $N/N\backslash(S|NP_{1,nom})$. The morpheme in (63) itself is a functor, so its arguments should be generated before generating the arguments of the anchor. First its last argument, $(S|NP_{1,nom})$, is sent to the generator. The generator attempts to find a lexical entry that unifies with this input. The verb *vur* 'kick' is found with this input.

(64) [vur],

```
((s,{(Tense,Num,Per,Gender),≤s-base},[]):(kick˜ball˜child,[]) |
(np,{(1,nom,Num1,Per1,Gender1),≤phrase},[]):(child,[]), |
(np,{(2,dat,Num2,Per2,Gender2),≤phrase},[]):(ball,[]),
phon(('vur',[])),
morph(('v',(free,concat),[]))
)
```

After this verb, its arguments should be generated. The object of the verb is a dative-marked *NP* ((*np*,{*(2,dat,Num2,Per2,Gender2),≤phrase*},[]):(*ball*,[])). With this input, the dative marker is found next. Again the input category and the result category of dative marker unify.

(65) [a,e,ye,ya],

```
((np,{(2,dat,Num,Per,Gender),≤phrase},[]):(ball,[]) \
(n,{(Index1,Case1,Num1,Per1,Gender1),≤n-num},[]):(ball,[]) phon(('(y)A',[])),
morph(('-DAT',(bound,affix),[]))
)
```

The dative case marker is a functor over *N*. In the next step, this *N* is generated, and put to the left of the dative marker since the categorial operator in the dative marker is backward slash.

(66)
```
((n,{(Index,Case,sg,3,Gender),≤n-root},[bar=0]):(ball,[])
phon(('top',[])),
morph(('n',(free,concat),[]))
)
```

At this point, the generation of the verb *vur* is complete. The category of the relativizer morpheme was $N/N\backslash\ (S|NP_{1,nom})$. We have generated $(S|NP_{1,nom})$ part, and the result is the phrase *top-a vur* 'kick the ball'. Since the operator before $(S|NP_{1,nom})$ is a backward slash, this phrase is put to the left of the relativizer morpheme. Hence, the morpheme *-an* is attached to the phrase *top-a vur* generating *topa vuran* 'the one that kicked the ball'. In a word-based approach, *-an* attaches to *vur* first. After this, the generator is called with the second argument of the relativizer morpheme:

(n,{(Index1,Case1,Num1,3,Gender1),≤n-base},[]):(child,[])

As a result of this call, the noun çocuG 'child' is found in the lexicon. Since the categorial operator before this argument in $N/N\backslash\ (S|NP_{1,nom})$ is forward slash, this lexical entry is added to the right of the relativizer morpheme.

(67)
```
((n,{(Index,Case,sg,3,Gender),≤n-root},[bar=0]):(child,[]) phon(('çocuG',[])),
morph(('n',(free,concat),[]))
)
```

Up to now, the phrase *topa vuran çocuk* 'the child who kicked the ball' is generated and the generation of the relativizer morpheme is complete. This phrase is the argument of the anchor (*uyu*). The category of the anchor was $S|NP_{1,nom}$. The neutral slash is first substituted with backward slash, therefore, the phrase *topa vuran çocuk* is put to the left of its functor, *uyu*.

(68) *top-a* *vur-an* *çocuk* *uyu-du*

ball-DAT kick-REL.SP child sleep-TENSE

'the child who kicked the ball slept.'

The generator attempts to find if there is another possible surface form for the input PAS. This time the neutral slash in $S|NP_{1,nom}$ is substituted with forward slash and the argument $NP_{1,nom}$ is put to the right of its functor.

(69) *uyu-du* *top-a* *vur-an* *çocuk*

sleep-TENSE ball-DAT kick-REL.SP child

The generator fails to find any more answers, therefore the generation process ends. Two surface forms (68, 69) have been generated for the input (60).

4.4 Discussion

Despite the fact that type-raised morphemes are put in the lexicon and used during parsing, they are not used during generation. During the generation process, only application rules of CCG (forward and backward application rules) are used, hence we can say that this is an 'application-based' approach. The consequence of this is that not all word orders of a sentence can be generated. Although all of the two possible orders of an intransitive sentence can be generated successfully, only four out of six possible permutations of a transitive sentence, and only eight out of twenty-four possible permutations of a ditransitive sentence can be generated[1].

The order of generation after the anchor always begins from the last argument. As a result of this, the verb and the direct object of a transitive sentence are always adjacent, since the category of a transitive verb is $S|NP_1|NP_2$. Hence, SOV (subject-object-verb), OVS, SVO, and VOS word orders of a transitive verb are generated, but the word orders in which subject is between object and verb, that is OSV and VSO, cannot be generated. Even if type-raised morphemes are put in the lexicon, the OSV and VSO word orders cannot be generated using the described algorithm. Because the category of type-raised accusative case suffix is:

(70) $((S{:}x\ y\ z\ |NP_1{:}z)/(S{:}x\ y\ z\ |NP_1{:}z\ |NP_2{:}y))\backslash N{:}y$

$((S{:}x\ y\ z\ |NP_1{:}z)\backslash(S{:}x\ y\ z\ |NP_1{:}z\ |NP_2{:}y))\backslash N{:}y$

[1] The surface forms in which the direct object is an indefinite direct object are not included in these figures.

When these type-raised categories are used, the subject can be placed between the verb and the direct object of a transitive sentence as well. Moreover, since there are several lexical entries for the same case suffix, overgeneration problem arises when type-raised categories are allowed in the lexicon. In addition, the generator may fail to terminate for some input. Because of these reasons, type-raised lexical entries are not taken into account.

During the lexical search, finding the correct case paradigm is an important issue. This is achieved by the use of hypocategories. Two inputs (71a-b) and their outputs (72a-b) from the generator are given below.

(71) a. *np-acc : cat*

kedi-yi

cat-ACC

b. *np-acc : POSS cat ayse*

Ayş-nin   kedi-si-ni

Ayşe-GEN.3 cat-POSS.3s-ACC2

The lexical entries for accusative case-marker morphemes are:

(72) a. ((np,{(2,acc,Num,Per,Gender),≤phrase},[]):(x,[]) \
(n,{(Index2,Case2,Num2,Per2,Gender2),≤n-num},[]):(x,[]) phon(('(y)I',[])),
morph(('ACC',(bound,affix),[]))
)

b. ((np,{(2,acc,Num,Per,Gender),≤phrase},[]):(x,[]) \
(n,{(Index2,Case2,Num2,Per2,Gender2),=n-poss},[]):(x[]) phon(('nI',[])),
morph(('ACC2',(bound,affix),[]))
)

The categories of *kedi* 'cat'(73a) and third person singular possessive marker (73b) are:

(73)

a. (n,{(Index,Case,sg,3,Gender),≤n-root},[bar=0]):(cat,[])

b. (n,{(Index,gen,Num,3,Gender),=n-poss},[]):(POSS x y,[])\
(n,{(5,gen,Num1,3,Gender1),=n-poss},[ref=def]):(x,[])\
(n,{(Index2,Case2,Num2,3,Gender2),≤n-num},[]):(x,[])

The incorrect *Ayşe-nin kedi-si-yi* is not generated since lattice condition in the hypocategories of ACC (*n-num*) and possessive marker (*n-poss*) should be equal because of the binary relation (=) in possessive marker. The incorrect *kedi-ni* is not generated because the hypocategory of ACC2 (*n-poss*) does not include the hypocategory of the noun *kedi* (*n-root*). Therefore these incorrect generations are prevented by the use of hypocategories.

The pro-drop characteristics of Turkish language is also reflected in the morphosyntactic generator. Whether subject and/or object of the sentence will be dropped is given as optional input. The dropping is performed if the will-be-dropped argument is a pronoun. Both subject and object can be dropped in the same sentence.

4.5 Comparison with Other Studies

Hakkani et al. (1996) proposed a tactical generation of Turkish in which the constituents may change their order according to the information structure of the sentences to be generated. In order to generate the different word orders indicated by the information structure, they have used a recursively structured finite state machine. The implementation of the system is based on the GenKit environment which was developed at Carnegie Mellon University - Center for Machine Translation. Case-frame feature structure is used to encode the contents of a sentence. Their generator takes content information where all lexical choices have been made as well as the information structure of the sentence where topic, focus, background of the sentence is specified. The information structure is used to select the appropriate word order. If no information structure is given, the 'default' word order of the sentence is produced as output. The morphology is handled outside the generation system. They use an independent morphological component which produces agglutinating surface forms. This component is expected to handle all aspects of morphology. These agglutinative forms are then used in tactical generator, therefore their system is word-based. For example, in the sentence in (74), their generator first generates *ver-diğ-i-ni*. It must find out that the verb *ver* 'give' is a ditransitive verb, so it should have three arguments. This is doneby deconstructing (or revealing) the stem of *ver-diğ-i-ni* (*ver*). Now, the verb's subcategorization allows the subordinate clause to pick up three arguments before the accusative *-ni* is attached to the phrase *Zeynep-in kedi-ye süt ver-diğ-i*. In our generator, the accusative marker is attached to the bracketed phrase without deconstruction, which in unbounded dependencies, can carry on indefinitely.

(74)	*Ahmet*	[*Zeynep-in*	*kedi-ye*	*süt*	*ver-diğ-i*	] *-ni*	*gör-dü*
	Ahmet.NOM	Zeynep-GEN.3	cat-DAT	milk.ACC	give-SUB-POSS.3s	ACC2	see-TENSE

'Ahmet saw that Zeynep gave the milk to the cat.'

Another study of Turkish surface form generation is by Hoffman (1994; 1995). She used Multi-set Combinatory Categorial Grammar formalism (Hoffman, 1992) which is an extension of CCGs for free word order languages. The subcategorization information associated with each verb does not specify the order of the arguments in the Multi-set CCG. Each verb in the lexicon is assigned a function category which specifies a *multiset* of arguments, hence it can combine with its arguments in any order. Each category is also associated with a semantic interpretation. She integrates this multi-set CCG formalism with a level of information structure that represents pragmatic functions, such as topic and focus. Each syntactic/semantic category is associated with an ordering category which bears linear precedence information. The ordering categories are assigned to lexical entries according to context-dependent word order restrictions found in the language. Using her formalism, Hoffman implemented a prototype database query system which generates Turkish sentences with context-appropriate word orders in answer to database queries. She adopted head-driven bottom-up generation algorithm for her generator. Her generator can deal with scrambling within embedded clauses as well as scrambling out of a clause. Again, morphology is not a part of her study. The entries in the lexicon should be fully inflected. However, Şehitoğlu and Bozşahin (1999) argue that putting all inflected forms of Turkish words in the lexicon is infeasible due to the recursive morphological processes such as the *-ki* relativizer.

The main difference in our study is the integration of morphology and syntax, and the use of a morpheme-based lexicon. There is no distinction between syntax and morphology, and morphemes are the main building blocks. Hence there is no need for rebracketing due to semantics either in parsing or generation.

4.6 Future Improvements

The main drawback of this study is that not every word order in Turkish is handled. The future work will focus on the generation of these word orders.

In a recent study, Bozşahin (2000) claims that Turkish is not an SOV language with scrambling, but an extraposition language with non-rigid word order, and it is verb-final in the lexicon. He searches evidence from the relationship between gapping and word order, and gives examples from Turkish data on gapping. He states that patterns of gapping originate from the lexicon rather than the grammar when directionality becomes a component of representation in the lexicon, which is the case in CCG-based approaches.

Baldridge (2000) argues that CCG formalism must be augmented in order to give a principal account of local scrambling. Addressing the issue of local scrambling, he proposes Set-CCG, an augmentation of CCG which handles local scrambling straightforwardly. Set-CCG, unlike Hoffman's (1995) Multiset-CCG, remains mildly context-sensitive. It accepts scrambled orders with a single category assignment by incorporating sets into categories and CCG rules. Set-CCG categories retain directional specification and Set-CCG rules retain a close relationship to their original counterparts. Baldridge also shows that Set-CCG and CCG are in fact strongly equivalent.

In the formalism if Set-CCG is used instead of CCG, local scrambling may be handled straightforwardly without type-raising and lexical ambiguity. In this case the lexical representation will be altered according to Set-CCG, and the generation algorithm will be modified to handle Set-CCG representation of the lexical entries. The basic word order of Turkish will be verb-final as Bozşahin proposed recently.

CHAPTER 5

CONCLUSION

In recent years, there is a growing tendency towards morpheme-based models. Şehitoğlu and Bozşahin (1999) propose a model in which inflectional morphology of Turkish is implemented in the lexicon by lexical rules. Keenan and Stabler (1997) present a generative approach that induces a natural algebraic notion of structure in which classical syntactic relations and certain morphological relations (e.g., case markers) are properly structural. Marslen-Wilson (1999) argues that the model of English organised in word-based units is incorrect and misleading, and it fails not only as a general model for the world's languages, but even as a model for English. He claims that even for English the basic unit of representation in lexicon should be morphemes. For a language like Turkish with a complex morphology, making the morphemes the unit of representation of lexicon is appropriate. The aim of this study was to generate Turkish surface forms using a morphemic lexicon.

This study used a categorial framework which claims that inflectional morphology is not different from syntax, so that they should be treated as one. The model is based on CCG, so it is a lexicalist formalism. In the lexicon, morphemes and words have the same representation in terms of syntactic, semantic, phonological, and morphological aspects.

The generation of Turkish surface forms using this morpheme-based categorial model was implemented in this thesis. The model correlates syntax, inflectional morphology, and semantics transparently. Generation of morphemes that have phrasal scope are not different from the generation of other morphemes since this is a morpheme-based generation. A semantic head-driven bottom-up generation algorithm was used in the generation. This algorithm is especially suitable for lexicalist approaches. Turkish surface forms were generated from a semantic interpretation, namely the PAS. The building blocks were the morphemes in the generation process, and only the forward and backward application rules of CCG were used.

REFERENCES

Baldridge, Jason M. 2000. Strong equivalence of CCG and set-CCG. ms., Edinburgh University.

Bar-Hillel, Y., C. Gaifman, and E. Shamir. 1960. On categorial and phrase structure grammars. *The Bulletin of the Research Council of Israel*, 9F:1–16.

Bouma, Gosse. 1988. Modifiers and specifiers in categorial unification grammar. *Linguistics*, 26:21–46.

Bozşahin, Cem. 1998. Deriving the predicate-argument structure for a free word order language. In *Proceedings of COLING-ACL*, pages 167–173, Montreal.

Bozşahin, Cem. 1999. Categorial morphosyntax: Transparency of the morphology-syntax-semantics interface. ms., METU.

Bozşahin, Cem. 2000. Lexicalism and word order identification: Evidence from gapping. ms., METU.

Bozşahin, Cem and Elvan Göçmen. 1995. A categorial framework for composition in multiple linguistic domains. In *Proceedings of the 4th International Conference on Cognitive Science of Natural Language Processing*, Dublin.

Calder, Jonathan, Mike Reape, and Henk Zeevat. 1989. An algorithm for generation in unification categorial grammar. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, pages 233–240.

Carpenter, Bob. 1997. *Type-Logical Semantics.* MIT Press, Cambridge, MA.

Şehitoğlu, Onur T. and Cem Bozşahin. 1999. Lexical rules and lexical organization: Productivity in the lexicon. In E. Viegas, editor, *Breadth and Depth of Semantic Lexicons.* Kluwer.

Dowty, David. 1988. Type raising, functional composition, and non-constituent conjunction. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures.* D. Reidel, Dordrecht.

Eisner, Jason. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 79–86.

Erguvanlı, Eser Emine. 1979. *The Function of Word Order in Turkish Grammar.* Ph.D. thesis, University of California at Los Angeles.

Hakkani, Dilek Zeynep, Kemal Oflazer, and İlyas Çiçekli. 1996. Tactical generation in a free constituent order language. In *Proceedings of the 8th International Workshop on Natural Language Generation*, Sussex, UK.

Hoffman, Beryl. 1992. A CCG approach to free word order languages. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 300–303. Student Session.

Hoffman, Beryl. 1994. Generating context-appropriate word orders in Turkish. In *Proceedings of the International Workshop on Natural Language Generation.*

Hoffman, Beryl. 1995. *The Computational Analysis of the Syntax and Interpretation of "Free" Word Order in Turkish.* Ph.D. thesis, University of Pennsylvania.

Joshi, Aravind. 1985. How much context-sensivity is required to provide reasonable structural descriptions: Tree-adjoining grammars. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Parsing: Psycholinguistic, Computational, and Theoratical Perspectives.* Cambridge University Press, New York.

Keenan, Edward L. and Edward Stabler. 1997. Bare grammar. In *Course Notes, European Summer School on Logic, Language, and Information.*

Marslen-Wilson, William. 1999. Abstractness and combination: The morphemic lexicon. In Simon Garrod and Martin J. Pickering, editors, *Language Processing.* Psychology Press, UK.

Moortgart, Michael. 1988. Mixed composition and discontinuous dependencies. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures.* D. Reidel, Dodrecht.

Oflazer, Kemal, Elvan Göçmen, and Cem Bozşahin. 1995. An outline of Turkish morphology. Technical report, Middle East Technical University.

Shieber, Stuart B. 1985. Restricting the weak generative capacity of synchronous tree adjoining grammar. *Computational Intelligence*, 8:333–343.

Shieber, Stuart M. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th International Conference on Computational Linguistics.*

Shieber, Stuart M., Gertjan van Noord, Robert C. Moore, and Fernando C. N. Pereira. 1989. A semantic-head-driven generation algorithm for unification-based formalisms. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics.*

Shieber, Stuart M., Gertjan van Noord, Robert C. Moore, and Fernando C. N. Pereira. 1990. Semantic-head-driven generation. *Computational Linguistics*, 16(1), March.

Slobin, Dan I. 1978. Universal and particular in the acquisition of language. In *Workshop on Language Acquisition*, University of Pennsylvania.

Steedman, Mark. 1985. Dependency and coordination in the grammar of Dutch and English. *Language*, 61(3):523–568.

Steedman, Mark. 1987. Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403–439.

Steedman, Mark. 1988. Combinators and grammars. In Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors, *Categorial Grammars and Natural Language Structures.* D. Reidel, Dordrecht.

Steedman, Mark. 1996. *Surface Structures and Interpretation.* MIT Press, Cambridge, MA.

Ümit Turan. 1995. *Null vs. Overt Subjects in Turkish Discourse: A Centering Analysis.* Ph.D. thesis, University of Pennsylvania.

van Noord, Gertjan. 1989. BUG: A directed bottom-up generator for unification based formalisms. In *Utrecht/Leuven working papers in Natural Language Processing.*

van Noord, Gertjan. 1990. An overview of head-driven bottom-up generation. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, Cognitive Science Series. Academic Press, New York, pages 141–166.

Vijay-Shanker, K. and D.J. Weir. 1993. Parsing some constrained grammar formalisms. *Computational Linguistics*, 19(4).

Weir, David and A.K. Joshi. 1988. Combinatory categorial grammars: Generative power and relationship to linear context-free rewriting systems. In *Proocedings of the 26th Annual Meeting of the Association forComputational Linguistics*, Buffalo, NY.

Williams, Edwin. 1981. On the nouns 'lexically related' and 'head of a word'. *Linguistic Inquiry*, 12(2):245–274.

Wittenburg, Kent. 1987. Predictive combinators. In *Proceedings of the 25th Annual Meeting of the Association for Computatinal Linguistics*, pages 73–79.

APPENDIX A

LIST OF ABBREVIATIONS

The meanings of the abbreviations used in the gloss are shown below.

n	Noun
v	Verb
pn	Pronoun
adj	Adjective
-PLU	Plural
-AGR	Agreement
-ACC	Accusative case
-DAT	Dative case
-ABL	Ablative case
-LOC	Locative case
-COM	Comitative case
-INST	Instrumental case
-COMP	Compounding
-COMP.POSS.3s	Compounding with agreement
-COMP2	Compounding
-COMP2.POSS.3p	Compounding with agreement
-GEN.1	Genitive case (1st person)
-GEN.2	Genitive case (2nd person)
-GEN.3	Genitive case (3rd person)
-POSS.1s	1st person singular possessive

-POSS.1p	1st person plural possessive
-POSS.2s	2nd person singular possessive
-POSS.2p	2nd person plural possessive
-POSS.3s	3rd person singular possessive
-POSS.3p	3rd person plural possessive
-NREL	Relative
-REL.SP	Subject relativization
-REL.OP	Nonsubject relativization
-SUB1I	Subordination (infinitive)
-SUB1G	Subordination (gerundive)
-SUB2I	Subordination (infinitive)
-SUB2G	Subordination (gerundive)
-TENSE	Tense
-PERS.1s	1st singular person

APPENDIX B

SAMPLE RUNS

The first line for each entry is the input to the generator. The second line is the output. The third line is the English gloss (not part of the output). We indicate number of solutions in parentheses in second lines. In some sample runs, all of the output surface forms are given.

(1) *s: sleep man*
Adam uyu. (2 results)
'Sleep man.'

(2) *s-past: sleep man*
Adam uyu-du. (2 results)
'The man slept.'

(3) *s-future: sleep man*
Adam uyu-yacak. (2 results)
'The man will sleep.'

(4) *s-narrative: sleep man*
Adam uyu-muş. (2 results)
'The man slept.' (reported speech)

(5) *s : read book man*
Adam kitab-ı oku. (6 results)
'The man read the book.'

(6) *s-past : read book man*

Adam kitab-ı oku-du. (6 results)

'The man read the book.'

(7) *s-past : give girl ball mehmet*

Mehmet kız-a top-u ver-di. (12 results)

'Mehmet gave the ball to the girl.'

(8) *s-past-1-sg-dropsub : forget ticket i*

Bilet-i unut-tu-m. (3 results)

'(I) forgot the ticket.'

(9) *s-past-dropobj : see she man*

Adam gör-dü. (2 results)

'The man saw (her).'

(10) *s-past-1-sg-dropsub-dropobj : forget she i*

Unut-tu-m. (1 result)

'(I) forgot (her).'

(11) *s|np-past : walk x*

Yürü-dü. (2 results)

'x walked.'

(12) *s/np-past-1-sg : sadden x i*

Ben üz-dü-m. (2 results)

'I saddened x.'

(13) *s-past : with(sleep man) ball*

Kadın uyu-du çocuk-la. (4 results)

'The woman slept with the child.'

(14) *np-loc : PLU house*

Ev-ler-de (1 result)

'At the houses'

(15) *n : POSS house i*

Ben-im ev-im (1 result)

'My house'

(16) *n : POSS book man*

Adam-ın kitab-ı (1 result)

'The man's book'

(17) *n : POSS child(POSS friend man)*

Adam-ın arkadaş-ı-nın çocuğ-ları

Adam-ın arkadaş-ı-nın çocuğ-u

'The man's friend's child'

(18) *n : POSS ball(POSS child(POSS friend man))*

Adam-ın arkadaş-ı-nın çocuğ-u-nun top-u (4 results)

'The man's friend's child's ball'

(19) *n : POSS(REL(at(ANA book)house)book)mehmet*

Mehmed-in ev-de-ki kitab-ı (1 result)

'Mehmet's book that is in the house'

(20) *n : PLU(COMP ticket(green bus))*

Yeşil otobüs bilet-leri (1 result)

'Green bus tickets' (bus is green)

(21) *n : green(PLU(COMP ticket bus))*

Yeşil otobüs bilet-leri (1 result)

'Green bus tickets' (ticket is green)

(22) *n : POSS(PLU(COMP ticket bus))man*

Adam-ın otobüs bilet-leri (1 result)

'The bus tickets that belong to the man.'

(23) *n : COMP(COMP rate(annual interest))(COMP card credit)*

Kredi kart-ı yıllık faiz oran-ı (1 result)

'Credit card annual interest rate'

(24) *n : COMP(COMP rate interest)(COMP card(annual credit))*

Yıllık kredi kart-ı faiz oran-ı (1 result)

'Annual credit card interest rate' (credit is annual)

(25) *n : annual(COMP(COMP rate interest)(COMP card credit))*

Yıllık kredi kart-ı faiz oran-ı (1 result)

'Annual credit card interest rate' (credit card interest rate is annual)

(26) *n : PLU(REL(at PRO(POSS(PLU house)we))PRO)*

Biz-im ev-ler-imiz-de-ki-ler (2 results)

'The ones that are in our houses'

(27) *n : REL(sleep(ANA man))man*

Uyu-yan adam (1 result)

'The man who sleeps'

(28) *n : REL(read book(ANA girl))girl*

Kitap oku-yan kız

Kitab-ı oku-yan kız

'The girl who read the book'

(29) *n : REL(at(ANA book)house)book*

Ev-de-ki kitap (1 result)

'The book that is at the house'

(30) *n : REL(see(REL(read(ANA book)child)book)(ANA man))man*

Çocuğ-un oku-duğu kitab-ı gör-en adam (1 result)

'The man who saw the book that the child read'

(31) *n : REL(read book (POSS child(ANA man)))man*

Çocuk-ları kitap oku-yan adam

Çocuğ-u kitap oku-yan adam

Çocuk-ları kitab-ı oku-yan adam

Çocuğ-u kitab-ı oku-yan adam

'The man whose child read the book'

(32) *n : REL(see(REL(read(ANA book)child)book)(ANA man))man*

Çocuğ-un oku-duğu kitab-ı gör-en adam (1 result)

'The man who saw the book that the child read'

(33) *n : REL(put(ANA chair)(REL(read(ANA book)man)book)mehmet)chair*

Mehmed-in adam-ın oku-duğu kitab-ı koy-duğu koltuk (1 result)

'The armchair that Mehmet put the book that the man read'

(34) *s-past : want(sleep(ANA child))child*

Çocuk uyu-ma-yı iste-di. (12 results)

'The child wanted to sleep.'

(35) *s-past : look(POSS car they) mehmet*

Mehmet onlar-ın araba-sı-na bak-tı (8 results)

'Mehmet looked at their car.'

(36) *s-past : give girl(REL(at(ANA book)house)book)(POSS friend man)*

Adam-ın arkadaş-ı kız-a ev-de-ki kitab-ı ver-di (16 results)

'The man's friend gave the book that is in the house to the girl.'

(37) *s-past : read(COMP book(REL(at PRO house)PRO))mehmet*

Mehmet ev-de-ki kitab-ı-nı oku-du. (4 results)

'Mehmet read the book that is at the house.'

(38) *s-past : at(kick ball(REL(look man(ANA mehmet))mehmet))house*

Adam-a bak-an Mehmet top-a vur-du ev-de. (8 results)

'Mehmet who looked at the man kicked the ball in the house.'

(39) *s-past : see(POSS(COMP ticket(REL(at(ANA car)house)car))girl)*
(REL(look(POSS child mehmet)(ANA man))man)

Mehmed-in çocuğ-u-na bak-an adam kız-ın ev-de-ki araba bilet-i-ni gör-dü (4 results)

'The man who looked at Mehmet's child saw the girl's car ticket in the house.'