AN ATN GRAMMAR FOR TURKISH

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Coşkun Demir

July, 1993

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

Asst. Prof. Dr. Kemal Oflazer (Advisor)

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

Asst. Prof. Dr. Halil Altay Güvenir

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

Asst. Prof. Dr. Cem Bozşahin

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray
Director of the Institute

ABSTRACT

AN ATN GRAMMAR FOR TURKISH

Coşkun Demir
M.S. in Computer Engineering and Information Science
Advisor: Asst. Prof. Dr. Kemal Oflazer
July, 1993

Syntactic parsing is an important step in any natural language processing system. Augmented Transition Networks (ATNs) are procedural mechanisms which have been one of the earliest and most common paradigms for parsing natural language. ATNs have the generative power of a Turing machine and were first popularized by Woods in 1970. This thesis presents our efforts in developing an ATN grammar for a subset of Turkish including simple and complex sentences. There are five networks in our grammar: the sentence (S) network, which includes the sentence structures that falls in our scope, the noun phrase (NP) network, the adverbial phrase (ADVP) network and finally the clause (CLAUSE) and gerund (GERUND) networks for handling complex sentences. We present results from parsing a large number of Turkish sentences.

Keywords: Natural Language Processing, Syntax, Parsing, Augmented Transition Networks, Turkish.

ÖZET

TÜRKÇE İÇİN BİR ATN GRAMERİ

Coşkun Demir
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Danışman: Dr. Kemal Oflazer
Temmuz, 1993

Sözdizimsel dil çözümlemesi herhangi bir doğal dil işleme sistemindeki önemli aşamalardan biridir. Genişletilmiş geçiş ağları (ATNs) doğal dil çözümlemesi için kullanılan ilk ve en yaygın örneklerden biridir. ATNs bir Turing makinasının üretici gücüne sahiptir ve 1970 yılında Woods tarafından kullanılıp tanıtılmıştır. Bu tez Türkçe'nin basit ve girişik cümleleri kapsayan bir altkümesi için bir ATN grameri geliştirilmesi çalışmalarımızı sunmaktadır. Gramerimizde beş tane ağ vardır: kapsamımızın içine giren cümle yapılarını kapsayan cümle (S) ağı, isim öbeği (NP) ağı, belirteç öbeği (ADVP) ağı ve son olarak girişik cümlelerin halledilmesi için tümcecik (CLAUSE) ve ulaç (GERUND) ağları. Sonuç olarak da yüksek sayıda Türkçe cümle çözümleme sonuçları sunulmaktadır.

Anahtar Sözcükler: Doğal dil işleme, dilbilgisi, çözümleme, ATNs, Türkçe.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my supervisor Asst. Prof. Dr. Kemal Oflazer for his guidance, suggestions, and invaluable encouragement throughout the development of this thesis. I would also like to thank to Asst. Prof. Dr. Halil Altay Güvenir and Asst. Prof. Dr. Cem Bozşahin for reading and commenting on the thesis. I owe special thanks to Prof. Dr. Mehmet Baray for providing a pleasant environment for study. I am grateful to the members of my family for their infinite moral support and patience that they have shown, particularly in times I was not with them.

Contents

List of Figures

Chapter 1

Introduction

Syntactic parsing of a natural language deals with the analysis of the relations between words and morphemes in a sentence and how they should be ordered to make structurally acceptable sentences in that language. In this thesis our aim is to parse the Turkish language using ATN formalism for representing grammatical knowledge.

Built upon Recursive Transition Networks (RTNs) [23], Augmented Transition Networks (ATNs) were one of the most common methods of parsing natural language in computer systems. ATNs have the generative power of a Turing machine, and unlike many other formalisms they are procedural. Owing to the convenience of developing an ATN grammar, they have been commonly used in a number of applications [19].

ATN grammars for a number of languages have been developed (e.g., for English see Winograd [23]). In this work, we present an ATN grammar for a substantial subset of Turkish which includes simple and complex sentences. Our system is able to find all syntactically correct parses of an input sentence. Since morphology plays an important role in syntactic parsing of languages like Turkish, our grammar uses the outputs of a two-level morphological analyzer developed for Turkish [1, 13]. It is this utilization that enables our grammar to use a large root word lexicon of about 23,000 roots words and increase the power of the system.

We have developed five networks for handling different syntactic components of the grammar. The first network is the sentence (S) network which parses a set of simple and complex sentence structures in Turkish. The second

is a noun phrase (NP) network including nominal and adjectival compounds in Turkish. The third is an adverbial phrase (ADVP) network including a subset of structures used as adverbial adjuncts in Turkish. The networks are interrelated as follows: The S network makes use of the NP and ADVP networks to parse its constituents. The NP network makes use of the CLAUSE network to handle participle and infinitive clauses and these clauses enable the S network include complex sentences. ADVP network makes use of NP and GERUND networks. Finally, the CLAUSE and GERUND networks are similar to S network and they make use of the NP and ADVP.

The outline of the thesis is as follows:

Chapter 2 includes an overview of natural language processing (NLP) and NLP applications together with a description of syntactic parsing. Chapter 3 contains an explanation of recursive and augmented transition networks. Chapter 4 presents an overview of the Turkish language and its syntax. This chapter is kept short in order to avoid unnecessary duplication of text and most of the features of Turkish syntax that influenced our implementation are also described in Chapter 5. Chapter 5 also includes descriptions of networks. Finally, a performance evaluation is made depending on the results of some test runs of the parsing system.

Chapter 2

Natural Language Processing

Natural Language Processing (NLP) is a research discipline at the juncture of artificial intelligence, linguistics, philosophy, and psychology that aims to *build systems capable of understanding and interpreting the computational mechanisms of natural languages.* Research in natural language processing has been motivated by two main aims:

- to lead to a better understanding of the structure and functions of human language,
- to support the construction of natural language interfaces and thus to facilitate communication between humans and computers.

The main problem in front of NLP which has kept it from full accomplishment is the sheer size and complexity of human languages. However, once accomplished, NLP will open the door for direct human-computer dialogs, which would bypass normal programming and operating system protocols.

There are mainly four kinds of knowledge used in understanding natural language: morphological, syntactic, semantic and pragmatic knowledge. *Morphology* is concerned with the forms of words. *Syntax* is the description of the ways in which words must be ordered to make structurally acceptable sentences in a language. *Semantics* describe the ways in which words are related to concepts. It helps us in selecting correct word senses and in eliminating syntactically correct but semantically incorrect parses. Finally, *pragmatic knowledge* deals with the way we see the world. Morphology, semantics and pragmatic

knowledge are out of our scope in this work, so they won't be described any further. For more information one can refer various books on natural language processing [7, 19, 23].

After a short description of NLP, we will first have a brief overview of NLP applications and then syntactic parsing of natural languages which falls into the scope of this thesis.

2.1 NLP Applications

In this section we will list some of the important areas for the application of natural language processing. For more examples of NLP applications the reader can refer to Winograd [23].

- **Machine Translation:** This is the first application area that aims to use a computer for translating text from one language to another language. There has been work in the area since the 1950s. Due to the difficulty of producing a high-quality, fully-automatic machine translator, human interaction should be used in translation. However, restricting the translation process to a specific domain makes the problem easier. The work done by Z. Sagay in 1981 in his master's thesis [16] and the study that is being done by K. Özgüven aim [14] to translate English text in Turkish.
- **Document Understanding and Generation:** A computer might read and "understand" documents, fitting their information into a larger framework of knowledge that can be used as abstractions of the document. The computer can then answer specific questions using this information. Document generation is a task related to document understanding that translates information stored in computer's memory in a formal language into natural language.
- **Natural Language Interface for Databases:** This is a question-answering system that carries on a dialog with a person in order to provide information from some stored body of knowledge. The knowledge can be stored in a huge database and these systems relieve the user of the need to be familiar with the database. Such systems can also include a generation component, producing natural language descriptions of what

is going to be read by the user of the system. This is one of the most developed areas in NLP.

- **Computer-Aided Instruction (CAI):** Computers have been used for education in many different ways, often involving some kind of question-answer interaction between student and computer. Integration of natural language in teaching specific subject domains certainly improves the power of a CAI system. Design of computer-aided education tools for teaching Turkish or any other language to foreigners can be an important application area for NLP.

- **Aids to text preparation:** Word processors are extensively used in the preparation and editing of texts. An advanced word processor can include *spelling checker* and *text critiquing* facilities. By looking up words in a stored dictionary and performing syntactic analysis of sentences, these systems can point out possible errors. For example the work done by A. Solak in 1991 presents the first spelling checker developed for Turkish [18].

There are many successful applications developed in the area since 1950s. We want to mention only two of them here: the SHRDLU system which made use of a procedural parsing scheme and LUNAR system which made use of ATNs as natural language front-end of the system.

SHRDLU, a system developed by Terry Winograd at MIT in 1971, was quite innovatory in comparison with other systems developed until that time and it embodied many important principles which have been taken up in later research. It was a system that could interact with a user about a world of toy blocks. One of Winograd's major contributions was to show that natural language understanding was possible for the computer in restricted domains. SHRDLU demonstrated in a primitive way a number of abilities. It was able to interpret questions, statements and commands, draw inferences, explain its actions and learn new words.

LUNAR system is also one of the successful works which provided access to a large database of information on lunar geology. LUNAR can be classified as an example of a natural language interface for a large database [25].

2.2 Syntactic Parsing of Natural Language

The analysis of a sentence involves assigning it a syntactic and a semantic structure. The syntactic structure deals with syntactic relations between the words and morphemes in the sentence while the semantic analysis deals with the meaning of the sentence. In this section we will concentrate on syntactic analysis.

Writing a grammar for the syntax of a natural language means collecting all the patterns for the sentences in that language that will be handled, and putting them down in one of the grammar-writing formalisms. In a big system, many rules are written. But with many rules, processing slows down, and it becomes hard to extend or debug the grammar. The processing slows down because of the nondeterminism in natural languages. Words can play different syntactic roles and have different meanings.

The grammars are written by computational linguists as rules that specify the ways in which words can be combined to form well-formed sentences. They require descriptively powerful, computationally effective formalisms for representing grammatical information or knowledge. A wide variety of formalisms have been employed in natural language processing systems, including context-free and context-sensitive phrase structure grammars, augmented transition networks, systemic grammar, lexical-functional grammar, generalized phrase structure grammar, and definite clause grammar. Winograd [23] provides a thorough account of these grammars and evaluates them in linguistic as well as computational terms.

It is important to distinguish between a *grammar* for a language and a *parser*. *Parsing* is concerned with machine processing of language. A sentence is considered to be parsed when each word has been assigned to a structure which is compatible with the grammar of the language. The parser takes a *grammar* and a string of words and gives either a grammatical structure imposed on that string of words, if the string of words is grammatical with respect to the grammar, or nothing, if it is not. Conceptually, the parser and the grammar are quite distinct: a grammar is simply an abstract definition of a set of well-formed structured objects, whereas a parser is an algorithm (a precise set of instructions) for arriving at such objects.

Recursive transition networks (RTNs) are one way of specifying grammars.

An RTN grammar consists of a collection of labeled networks. The networks themselves consist of a collection of states connected by directional arcs labeled with the names of syntactic categories.

RTNs themselves have been overshadowed in NLP by an elaborated version of the formalism known as the augmented transition network (ATN). An ATN is simply an RTN that has been equipped with a memory and the ability to augment arcs with actions and conditions that make reference to that memory. During the 1970s, augmented transition network (ATN) grammars [24] were widely used in natural language processing systems. The ATN formalism embodies a model of language recognition or parsing as a process of traversing arcs between states in a network. The ATN formalism is both computationally powerful and inherently procedural. The work done in this thesis depends on the ATN formalism and ATNs will be investigated in detail in the following chapter.

Chapter 3

Augmented Transition Networks

ATNs are procedural mechanisms which are built upon Recursive Transition Networks (RTNs) with the addition of certain augmentations. In this chapter we will first have a look at the history of Augmented Transition Networks (ATNs). We will then investigate Recursive Transition Networks (RTNs) as predecessors of ATNs. We will then build the ATN formalism upon RTNs with explanations of the augmentations. Finally, we will end the chapter with a discussion on a comparison of ATNs with declarative formalisms.

3.1 History of ATNs

Chomsky's transformational grammar which was developed in the 1960s consisted of a set of transformations that could be used to derive more complex sentences from simpler ones [4]. Unfortunately, it proved computationally infeasible to undo or reverse these transformations in a principled way, for building a language analyzer that would take arbitrary sentences and understand them in terms of their source representations. ATNs were developed as a programming language for writing analyzers where the undoing of transformations could be carried out during processing [7].

The idea of a transition network parsing procedure for natural language was originally suggested by Thorne et al. [20], and was subsequently refined in an implementation by Bobrow and Fraser [3]. ATNs were first popularized by Woods [24] where he used them as the natural-language front end to a system for accessing geological data on the Apollo lunar samples. Later Kaplan worked

with Woods' version of ATNs and argued their psycholinguistic plausibility [9].

Despite some of its drawbacks that we will discuss at the end of this chapter, ATNs remain one of the most successful parsing strategies yet developed. Since the early use of the ATN in the LUNAR system [25], the mechanism has been exploited in many language-understanding systems. From that time on ATN grammars for various languages have been developed (e.g., for English refer to Winograd [23] for a large ATN grammar or to Noble [12] for a simpler one). One of the recent examples of the use of ATNs for generation is the study by Shapiro [17].

To the best of our knowledge, the work done in this thesis is the first attempt to develop a large-scale ATN grammar for Turkish.

3.2 Recursive Transition Networks

Recursive transition networks (RTNs) allow us to deal naturally with some of the recursive structures in natural languages. RTNs are formally equivalent in power to a context-free grammar and they provide a foundation on which to build Augmented Transition Networks [23].

RTN grammars consist of a set of labeled networks consisting of a finite set of nodes (denoting states) connected by labeled directed arcs. The arcs are labeled with a *word*, a *lexical category* or a *syntactic category* that is the label of some network in the grammar. The fundamental difference of RTNs from finite-state transition networks (FSTNs) is the extra concept of a *named subnetwork*. That is, it is possible for an arc to name a subnetwork to be traversed. The idea is that if we have a commonly used bunch of arcs, we can express this abstraction by making it into a self-contained, named network. This network can then be referenced by its name in a network that needs it, rather than having to appear expanded out in every place of the grammar [7].

Although introducing named subnetworks seems conceptually like a small change to FSTNs, it does introduce significant complexity in the procedure for traversing a network. The basic problem is that, to traverse one arc, it may be necessary to traverse a whole subnetwork. While that subnetwork is being traversed, the position of the original arc must be remembered, so that the traversal can resume there afterwards. RTN can be regarded as a specification

of a pushdown automaton (PDA). Informally, in an RTN, to traverse an arc that is labeled with a subnetwork name instead of a word or lexical category, it is necessary to traverse the subnetwork named, but remembering where to resume when that has been done. A pushdown automaton is equipped with a stack that can be used for this purpose.

To determine whether a given string of words is grammatical according to an RTN, it is necessary to find a route, starting at initial states and ending up at final states. At each stage, progress can only be made by following the arcs in the network, whose labels indicate which categories the successive words in the string must belong to. Only if all these conditions can be met, has a successful path have been found.

RTNs can be used in place of rules to define a language. There is a set of networks in an RTN grammar, and the number of networks is determined according to the complexity of the language. A network consists of a set of *states (nodes)* connected by *directed arcs*. Some states can further be designated as *initial states* and as *terminal states* to denote the entry and exit states of the network. An arc represents an allowable transition from the state at its tail to the state at its head, the label indicating the input symbol which must be found in order for the transition to occur. A network is not used to store information about a particular parse. It represents a pattern to be matched against potential sentences. Transition networks are represented graphically with circles for states and arrows connecting the circles representing arcs.

The types of arcs in an RTN vary according to their labels. There are four types of arcs are used:

1. **Category Arcs :** The most frequently used arcs are *category* arcs where the label is a *lexical category*. If the label on the arc matches with the lexical category of the first word of the input tape, the state change of the arc is performed and the first word of the input tape is consumed. The arc NP-3 in Figure 3.1 is an example of a category arc where the label is a "Noun."

2. **Word Arcs :** The label of the arc is a word itself. It is suitable for words which have a specific and constant syntactic function in the grammar of the natural language that is considered. Since there is a single word consumption from input tape, this arc type is similar to category arcs.

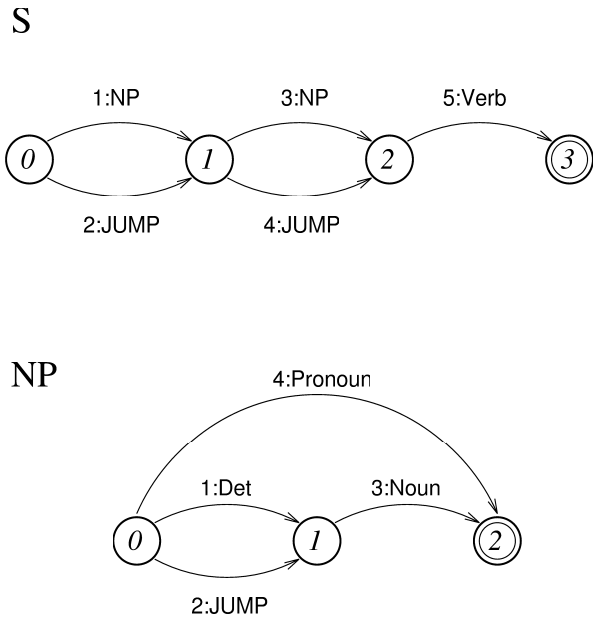

Figure 3.1. Simple RTN Grammar for Turkish

3. **Jump Arcs :** This is a very special arc that allows a transition in the grammar with possible actions, but without advancing the input tape. It is useful for bypassing optional grammar elements. This is equivalent to a null-transition in FSR and PDA formalisms. Jump arcs do not change the formal power of the networks, but it makes it more convenient to write them. The arc S-2 in Figure 3.1 is an example of a jump arc which denotes that arc S-1 is optional.

4. **Push Arcs :** When the label is the name of a network, the parser pushes the current state to a stack and switches to the other network. The tape consumption of push arcs is determined according to the network that is the label of the arc. If the network fails, the arc is rejected and no consumption is made. If it succeeds the arc is matched against one or more words. The arc S-1 in Figure 3.1 is an example of a push arc for parsing subject of the sentence using network NP.

The RTN grammar in Figure 3.1 is an example for handling a very small subset of Turkish sentences. The Noun Phrase (NP) network in the figure accepts the structures *Determiner+Noun, Noun,* or *Pronoun.* The network accepts these structures and returns them back to the place where NP network is pushed from the Sentence (S) network.

The simple S network is for handling the *Subject+Accusative Object+Verb*

structures of Turkish sentences. Since the subject and object of the sentence are optional with the use of Jump arcs other combinations such as *Subject+Verb*, *Accusative Object+Verb* and *Verb* are also accepted. The problem with this network is that, it can not check any case information because of the limitations of the RTN formalism. In Turkish the only criteria that distinguishes subject from an accusative object is the case information. Subject is nominative and object is accusative. Since the RTN formalism has no memory and no built-in structure that allows checks on structures that are accepted by arcs, these case controls can not be done with RTNs.

Another problem with this example grammar, is that transitivity of the verb can not be checked. Since, intransitive Turkish verbs can not take object, this should be handled by any grammar that is written for Turkish. However, this is not possible with RTN formalism because of the previously mentioned problems. Two example sentences that are parsed as valid by the above networks are as follows: In the sentence *Ben bu okulu sevdim. (I liked this school.)*, the verb is transitive, and it can take an accusative object and hence the sentence is valid. In another example sentence *Ben okulu geldim. (I came the school.)*, the verb *gelmek (to come)* is intransitive, and can not take an accusative object, but it is accepted by the grammar above.

We need a more powerful parsing mechanism for natural languages, since RTNs are inadequate for this purpose.

3.3 Augmented Transition Networks

Augmented Transition Networks (ATNs) are procedural mechanisms with the generative power of a Turing machine [9, 23], that were built upon RTNs in the 1970s.

The addition of conditions and actions to the arcs of network and the use of registers are the extensions in ATNs not originally available in RTNs. An ATN can contain two different arcs with the same label, starting and ending states but with different conditions and actions.

The extensions in ATNs and their usage are as follows:

1. **Registers :** Registers are similar to the variables of a programming language which allow values to be remembered during a network traversal. They are used as a storage for keeping features of the current state of the parse.

   There are two types of registers in an ATN. The first type of registers are user-defined registers which are local to the network they are defined for. These registers should be manipulated carefully by the writer of the grammar. The assignments of these registers are done by the grammar writer in the *actions* part of the arcs.

   In addition to the user-defined registers local to the network, there are two global registers. The first one called *Star* automatically holds the value that we are currently considering. In other words, for a *category* arc *Star* keeps the word itself and in case of a *push* arc, it keeps the structure returned from the network. *Hold* is the other global register which is used for handling long-distance dependencies. It is implemented similar to *Star* with one exception. The contents of the *Hold* register can be changed by the grammar writer, but *Star* register can not.

2. **Conditions :** Conditions are the restrictions under which an arc can be taken. A condition is a Boolean combination of predicates involving the current input symbol (kept in *Star* register) and local register contents. An arc can not be taken if its condition evaluates to false (symbolized by NIL), even though the current input symbol satisfies the arc label. This means first, that more strict restrictions can be imposed on the current input symbol than those conveyed by the arc label, and second, that information about previous states and register structures can be used to determine future transitions.

   The condition predicates can be arbitrary functions in LISP notation. For an arc to be taken these conditions are checked first and then if they are satisfied the state change is performed and finally the operations in the *actions* part of the arc are performed.

3. **Actions :** Actions perform register assignment and structure-building operations. The current state of the parse is changed in this part by the change in local and global registers and this state is used in the following arcs.

The arc types described in the section for RTNs are used in ATNs. The

```
Augmentations for
Sentence (S) network :

Registers:

Subject, Object, Predicate,

Conditions and Actions:

S-1: NP-   Subject
     C: Case(*) is NOMINATIVE
     A: Set Subject to *.


S-3: NP-   Accusative Object
     C: Case(*) is ACCUSATIVE
     A: Set Object to *.


S-5: Verb- Transitive Verb or
           Intransitive Verb
     C: (Object <> nil &
         * is intransitive)
              or
        (Object exists &
         * is transitive)
     A: Set Predicate to *.
```

```
Augmentations for
Noun Phrase (NP) network :

Registers:

Agr (Agreement), Poss (Possessive),
Definite, Case, Res (Result),
Demonstrator,

Conditions and Actions:

NP-1:DEMONS
     A: Set Definite to True.
        Set Demonstrator to *.

NP-2:JUMP

NP-3:NOUN
     A: Set Case to case of *.
        Set Poss to possessive of *.
        if (Poss <> nil or
            Demonstrator exists)
           Set Definite to True
        endif.
        Set Agr  to agreement of *.
        if Demonstrator
           Set Res to
              (list Demonstrator *)
        else
           Set Res to *
        endif.

NP-4:PRONOUN
     A: Set Definite to True.
        Set Case to case of *.
        Set Poss to possessive of *.
        Set Agr  to agreement of *.
        Set Res to *.
```

Figure 3.2. Augmentations for simple ATN for Turkish

addition of conditions and actions to these arc types in our implementation of the ATN parser are shown below:

- (CATEGORY <CATEGORY NAME> <CONDITION> <ACTIONS>)
  In CATEGORY arcs the dictionary entry of the first word on the input string is checked for the presence of <CATEGORY NAME> such as noun, verb,etc. If the word has that category, then <CONDITION> is tested. If <CONDITION> holds then the <ACTIONS> are performed and finally the state is changed.

- (WORD <WORD> <CONDITION> <ACTIONS>)
  WORD arcs check the first word in the input string against <WORD> After this step, they are handled the same as CATEGORY arcs.

- (JUMP <CONDITION> <ACTIONS>)
  JUMP arcs are also similar without any word consumption.

- (PUSH <CONDITION1> <NETWORK> <CONDITION2> <ACTIONS>)
  For PUSH arcs, first the <CONDITION1> which include tests on the current state of the parse is tested. If it is satisfied then the control is passed to the <NETWORK> after pushing the current state. Upon popping from the <NETWORK>, <CONDITION2> which include tests that should be applied on the structure returned from the <NETWORK> is tested. If <CONDITION2> holds then the <ACTIONS> are performed followed by a state change.

The arc types given above are available in any ATN implementation. New arc types can be defined according to the needs of the grammar writer. The reader can refer to [19, 23] for implementation of larger sets of ATN arcs.

The simple RTN for Turkish can be converted to an ATN by the addition of the augmentations in Figure 3.2. The conditions and actions that are added solve the problems that are mentioned in the previous section.

3.4 A Comparison of Procedural and Declarative Formalisms

Computer programming is the activity of giving a computer a precise set of instructions for how to perform some task. Certainly, a lot of knowledge that humans have seems to be represented in this *procedural way.* Another way is to represent the rules and principles themselves *declaratively* as symbolic structures to be manipulated by the program.

ATN-based parsers were probably the most common kind of parser employed by computational linguists in the 1970s, but they have begun to fall out of favor in recent years. ATNs have proved to be very useful in a variety of language understanding systems, but they have some drawbacks:

1. RTN part of the grammar (networks) have a declarative nature and easy to understand, but the additional augmentations destroy the declarative nature of the formalism.

2. They can be very expensive to run if a great deal of backtracking is required. When backtracking, a lower level constituent may be parsed repeatedly always yielding the same result. However, adaptation of chart parsing to ATNs solves this problem [7, 23].

3. Unless all the words in the sentence are known to the system and the entire structure of the sentence matches exactly a path in the network, the parsing process will fail. There is no ability to perform partial matching. It also provides less help as to where the problem lies in the sentence.

4. ATN grammar development is hindered by lack of modularity. Changes in one part of the grammar may have unforeseen and unwanted side-effects elsewhere.

5. Although semantic information can be used to reject possible paths by incorporating it into tests on the arcs, it is not easy to use such knowledge to help choose the most likely of several possible paths so that it can be explored first. There is no way to use heuristic functions.

Because of the above problems, people have more recently turned their attention towards *declarative* formalisms for specifying grammars. In contrast

to *procedural* formalisms, *declarative* formalisms can be understood without reference to underlying models of language processing.

But, despite these drawbacks, the ATN remains a very useful mechanism. It has been exploited in many language-understanding systems.

Chapter 4

The Turkish Language

Turkish is a member of the south-western or Oghuz group of the Turkic family of languages, which extends over a vast area in southern and western Siberia and adjacent portions of Iran, Afghanistan and China, Anatolia, Balkans, Cyprus and Middle East [10, 18]. The subject of this study is the official and literary language of the Republic of Turkey.

In this chapter, we will not deal with all aspects of Turkish grammar. We will only investigate in some depth the syntax of Turkish which deals with how the words are arranged into phrases and sentences. Syntax is an important part of Turkish grammar. Other issues of syntax will be discussed in the following chapter together with our implementation in order to avoid unnecessary duplication.

4.1 The Syntax of Turkish

Turkish is a predominantly subject-object-verb (SOV) language, however the order of phrases may be changed to emphasize certain constituents of the sentence. Since the position of emphasis in Turkish is the position immediately before the predicate, the constituent which is considered as important is put closer to the predicate of the sentence. This is called as the *placements of constituents* rule. The variations of sentence *Onur dün otobüsle Ankara'ya gitti. (Onur went to Ankara by bus yesterday.)* which is considered as the usual sequence can be written with following variations:

- Onur dün otobüsle Ankara'ya gitti. (Usual sequence)
- Onur dün Ankara'ya *otobüsle* gitti. (Instrumental is emphasized)
- Onur otobüsle Ankara'ya *dün* gitti. (Time is emphasized)
- Dün otobüsle Ankara'ya *Onur* gitti. (Subject is emphasized)

Some of the important features of Turkish syntax can be listed as follows:

1. In Turkish syntax secondary constituents come before primary constituents. This is an important criteria which distinguishes Turkish syntax from others.

2. Elliptical expressions have an important use in Turkish. Sentences with *covert subject* and compounds without modifiers are available in Turkish and they are not available in most of the other languages. Examples are *Konuştum. (I spoke.)* with a covert subject *ben (I)* and *evim (my house)* with a covert modifier *benim (my).*

3. Turkish is an *agglutinative language* in which syntactic relations between words or concepts are expressed through discrete suffixes. As the suffixes play an important role for doing syntactic analysis of Turkish, morphological analysis is very important. For example, in English cases of nouns (e.g., ablative, dative, locative) are constructed with the assistance of separate words (prepositions like from, to, at) and these words are used to bind adjuncts to verbs. In Turkish, cases of nouns are obtained with the attachment of suffixes to words. There are six cases of nouns: The simplest form, with no suffixes is the *absolute (nominative) case.* The *accusative case*, with suffix "+yI" [1], marks the definite object of a verb. The *genitive case*, with suffix "+nIn" denotes possession. The *dative case*, with suffix "+yE" [2] denotes the indirect object of a verb and the end of motion. The *locative case*, with suffix "+dE", denotes the place of action. Finally, the *ablative case*, with suffix "+dEn" denotes the point of departure.

4. Turkish is a *head-last* language. English has prepositions, which precede the noun to which they refer; Turkish has postpositions, which follow the

[1] "I" is used to denote high vowels ı,i,u and ü.
[2] "E" is used to denote low-unrounded vowels e and a.

noun. The example *Mehmet için (for Mehmet)* shows this feature where the postposition *için (for)* is used.

The main topic of syntax is the *sentence*. A sentence is produced with the combination of constituents with different tasks. We will describe these constituents in the next section.

4.1.1 Constituents of a Turkish Sentence

There are two main constituents in a sentence. They are *subject* and *predicate*. Among them *predicate* is obligatory, whereas *subject* may be covert as mentioned previously. Other constituents of a Turkish sentence are *objects* and *adjuncts*. The usage of these constituents is optional and dependent on the properties of verb [5, 10].

4.1.1.1 Predicate

In Turkish syntax, in addition to verbs, other lexical categories can also function as predicates with the addition of auxiliary verbs which are forms of the verb "to be." These forms are: *idi (definite past), imiş (inferential), ise (conditional)*. The present tense of present tense is obtained by the attachment of personal suffixes except the third person where the copula *"+dIr"* is used. In addition to the lexical categories like noun, adjective, pronoun, adverb,etc, compounds can also be used as predicates of *nonverbal (nominal) sentences*. The *verbal sentences* are sentences in which the predicate is a verb.

Predicate is found as the last constituent in a usual sequence of a Turkish sentence. However, in *inverted sentences*, there is no fixed place for the predicate.

4.1.1.2 Subject

Subject is the second main constituent. The reason it comes after the verb is that a verb can produce a sentence itself with a covert subject, but a subject can not do the same thing without a verb. Subject can be found in the sentence

by using the questions **who?** or **what?**. In regular sentences subject comes before verb. Turkish subject is always in nominative case.

The subjects of active-verbal and nonverbal sentences can be covert or existent in a sentence. In sentences where the predicate is a passive verb, the indefinite nominal object is considered as subject although it is not the agent of the action of the verb in the sentence. This subject is called *supposed subject (sözde özne)* in Turkish syntax. In the sentence *Kapı açıldı. (The door is opened.)*, the word *Kapı (The door)* is the supposed subject. If there is no indefinite nominal object in such a sentence, the subject is considered as covert.

Subject has no fixed place in sentence according to the *placements of constituents* rule mentioned before. However, in nonverbal sentences subject generally comes immediately before predicate.

- **Subject-Predicate Agreement in Turkish:** It is classified as *number* and *person* agreement. In *number* agreement, the general rule is that if the subject is singular, then the predicate is singular, if the subject is plural, then the predicate is plural. However, this rule is violated by many exceptional cases which allow plural subjects to be used with singular predicates and vice versa. For example, a singular verb is commonly used with an inanimate plural subject, a plural verb may be used with an animate plural subject representing a number of people acting as one, or a plural verb may be used with a singular subject, second or third person, for a mark of respect. The examples to these cases are: *atlar koştu. (The horses ran.), Adamlar geldi. (The men came.), Taşlar aşagı düştü. (The stones fell down.). Person* agreement is more strict. If there is one subject in the sentence, person of the subject should be equal to the person of verb. In the valid sentence *Ben geldim. (I came.)* the subject and predicate are both first person singular, whereas in the invalid sentence *Ben geldi.* the subject is first person singular and predicate is third person singular.

4.1.1.3 Object

Object is the third important constituent of the sentence. These are further classified as *direct* and *indirect* objects:

1. A *direct object* can be either in nominative or accusative case. Nominative objects are called *indefinite objects*. Because of indefiniteness, definite structures like proper nouns, first and second person pronouns, definite nominal and adjectival compounds can only function as definite objects in the sentence. Indefinite objects do not allow any other constituent to come in between predicate and itself. Accusative objects are called *definite objects*. Their place of occurrence is flexible and there can be other constituents between predicate and this type of object.

   The usage of direct objects in a sentence is determined according to the transitivity of the verb. In transitive-verbal sentences object is an obligatory element. They may be omitted from sentence in some cases, but the object is considered as existent in these cases also. Direct objects are totally non-existent in nonverbal sentences and intransitive-verbal sentences. In transitive-passive-verbal sentences indefinite object can function as *sözde özne (supposed subject)* as mentioned before. There can not be two direct objects in a sentence from different types.

2. An *indirect object* can be in dative or ablative cases. Dative indirect objects indicate the person (or thing) to or for whom the action is directed. Ablative indirect adjuncts indicate the person (or thing) from whom the action proceeds.

   Verbs may or may not take indirect objects according to their argument structure information. For example the verb *vermek (to give)* can take a dative indirect object as in the example *Adama yemek verdim. (I gave the man food.)*. The verb *almak (to take)* can take an ablative object as in the example *Kitabı Onur'dan aldım. (I took the book from Onur.)*. An indirect object can be in locative case very rarely. For example, the verb *ısrar etmek (to insist)* can take a locative indirect object as in the sentence *bu konuda ısrar edeceğim. (I will insist on this subject.)*.

4.1.1.4 Adjunct

Adjuncts are further classified as *indirect* and *adverbial* adjuncts:

1. **Indirect Adjuncts (Oblique Objects) :** These modify the meaning of verb by specifying the place, direction or source of the action. These adjuncts are also classified as *oblique objects* in some grammar books. We

will use both names interchangeably in the rest of this thesis. There are three types of indirect adjuncts, classified according to their case. *Dative* indirect adjuncts indicate the place to or toward which the motion is directed, *locative* indirect adjuncts indicate the place at which an action occurs and *ablative* indirect adjuncts indicate the place from which motion proceeds.

Indirect adjuncts generally precede the predicate in verbal sentences with intransitive verbs. In sentences with transitive verbs, they precede object because object's place is of higher priority. The usage of indirect adjuncts is highly dependent on the argument structure of verbs. Locative indirect adjunct can be used in all types of verbal and nonverbal sentences. Dative and ablative indirect adjuncts can not be used in nonverbal sentences and their usage in verbal sentences is dependent of the verb. Some verbs require dative, some require ablative and some of them require both of them.

2. **Adverbial Adjuncts :** Adverbial adjuncts modify and strengthen the meaning of verb from mainly time, direction, quantity and quality aspects. There are many structures used as adverbial adjunct in Turkish. Examples of temporal adverbial adjuncts are: *sabahleyin (in the morning)*, *bir haftadan beri (since one week)*, *ben okula giderken (while I was going to school)*. Examples of directional adverbial adjuncts are: *eve doğru (towards home)*, *kapıdan içeri (through door)*, *içeri (inside)*, *yukarı (upwards)*, *geri (backwards)*. Examples of qualifying adverbial adjuncts are: *okumak için (for reading)*, *sabırsızlıkla (impatiently)*, *kesinlikle (certainly)*, *ancak (barely)*, *otobüsle (by bus)*. Examples of quantifying adverbial adjuncts are: *az (few)*, *biraz (some)*, *çok (very)*, *daha çok (further)*.

   Adverbs, adjectives and gerundive clauses are frequently used in adverbial adjuncts. The usage of nouns is restricted to temporal nouns only and the pronouns are not used in adverbial adjuncts.

   The place of adverbial adjuncts in the sentence is determined according to type of adjunct. Temporal adverbial adjuncts mostly occur in the beginning of the sentence. Directional and quantifying adverbial adjuncts are found closer to the predicate. They are generally found as the second constituent in a regular sentence.

4.1.2 Sentence Types

Turkish sentences can be classified according to various criteria. In this section we will discuss these classifications.

The main constituent of a Turkish sentence is the verb. Therefore, the first classification of Turkish is done according to the type of verb. There are two types of sentences classified according to type of predicate:

- **Verbal Sentences :** The predicate of a verbal sentence is verb. They are further classified according to the transitivity of the verb and according to whether the verb is active or passive. According to this classification, there are *transitive-active verbal* sentences, *transitive-passive verbal* sentences, *intransitive-active verbal* sentences, and *intransitive-passive verbal* sentences.

  In a verbal sentence, the occurrence of subject, object and predicate is restricted to one, whereas indirect and adverbial adjuncts can occur more than once in different forms. An object's existence further depends on the transitivity of the verb.

- **Nonverbal(Nominal) Sentences :** In a nonverbal sentence, a noun, an adjective, an adverb, a nominal or an adjectival compound can function as predicate. The forms of the verb "to be" are used to convert these nonverbal structures to nominal predicates as mentioned previously. Among the forms of the verb "to be" the conditional *ise* can not be used as a predicate of the main sentence. It can only be used as a predicate of a clause as in the example: *Hepimiz hazırsak yola çıkalım. (If we all are ready, let's depart.).* The negative of "to be" is obtained by putting after *değil (not)* following the nonverbal structure. They together function as a predicate of the nonverbal sentence.

  In a nonverbal sentence, the object is non-existent and usage of indirect adjuncts is limited with locative indirect adjuncts. Adverbial adjuncts can be used with no limitation.

Another important criteria for sentence classification of Turkish is word-order. There are two types of sentences classified according to this criteria:

- **Regular Sentences :** The predicate is found at the end of the sentence.

The reason that they are called regular is that they obey the general rule of Turkish syntax which says that secondary constituents should precede the primary constituent in a sentence.

We can list general patterns for Turkish regular sentences according to type of predicate as follows:

- **With transitive-verbal predicates :**
  Subject + Adverbial Adjunct + Indirect Adjunct + Object + Predicate
- **With intransitive-verbal predicates :**
  Subject + Adverbial Adjunct + Indirect Adjunct + Predicate
- **In nonverbal sentences :**
  Indirect Adjunct + Adverbial Adjunct + Subject + Predicate

- **Inverted (Devrik) Sentences :** The verb is not at the end of the sentence. The reason these sentences are called inverted is that verb which is the obligatory element is not found in its usual place.

Sentences are further classified according to their structure. The types of sentences according to structure are :

- **Simple Sentence :** There is only one independent judgement in the sentence. Simple sentences are not suitable for expressing complex thoughts, events or situations.

- **Compound Sentence :** In a compound sentence there are secondary judgements along with the main judgement. Compound sentences are further classified according to their dependent clauses. There are *complex* sentences where the dependent clause is a participle, infinitive or a gerund clause, *conditional* sentences where the main clause and dependent clause are integrated by a condition and finally there are compound sentences (*kaynaşık* sentences in Turkish) where the dependent clause is a *substantival sentence.* A substantival sentence is a complete sentence that can function as a noun clause or adjectival clause within a longer sentence. Among them complex sentences fall into our scope and they will further be described in the following chapter.

- **Ordered Sentence :** Two or more complete sentences can be combined to form an *ordered* (*sıralı* in Turkish) sentence. These types of sentences are out of our scope.

There is also a classification of sentences according to their semantics. There are *positive, negative* and *interrogative* sentences. They will not be described here.

Chapter 5

Implementation

The scope of our work is the design and implementation of an ATN grammar for a subset of Turkish. This subset is carefully chosen to cover a wide range of Turkish sentences structures.

Figure 5.1 shows the general structure of our implementation. The ATN parser is the tool that makes use of the morphologically analyzed words, network definitions and arguments of verbs to decide whether the input sentence is syntactically correct or not. If the sentence is found to be correct, the parser produces outputs for all ambiguous parses of it. There is a simple user interface that has the responsibility of feeding the parser with sentences and printing the output of the parser in a suitable format.

We will explain in some detail the ATN parser, network definitions, and arguments of verbs later in this chapter. For the morphological analysis and Turkish lexicon one can refer to Oflazer [13] if more information is needed.

The current version of grammar includes an S network which includes frequently used simple and complex sentence structures of Turkish. The network makes use of two other networks: NP and ADVP. The NP network is the most commonly used one and is called recursively by both itself and ADVP network. NP network makes use of CLAUSE network for handling participle and infinitive clauses. ADVP network in turn makes use of a GERUND network for handling gerund clauses.

In this chapter we will first quickly scan through the ATN parser that we used and then we will explain in some detail the overall ATN architecture with

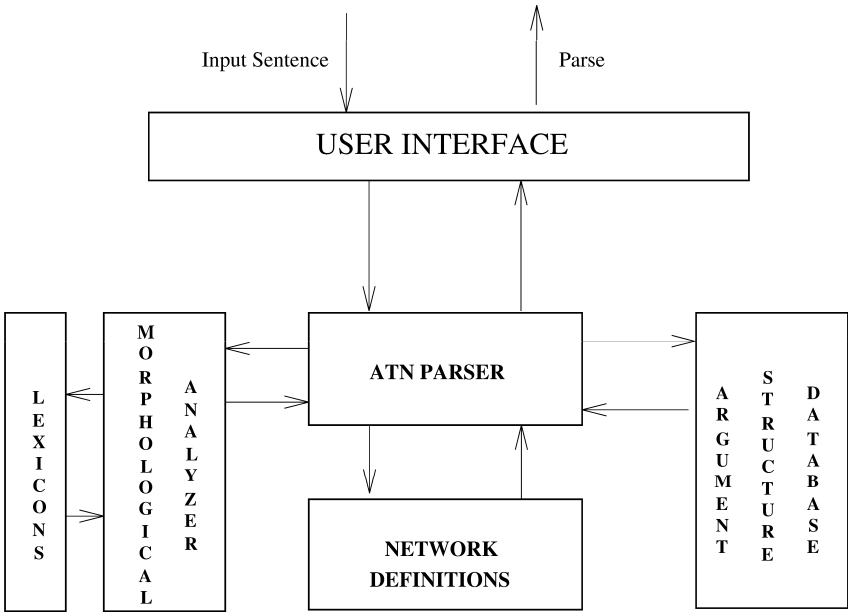

Figure 5.1. System Architecture

network definitions and the structures that the networks accept.

5.1 The ATN parser

An ATN parser is a formalism which can determine whether a sentence conforms to the constraints of the syntax of a grammar, and also can build a representation of the syntactic structure. The ATN parser that is used in this work is a top-down left-to-right parser that uses depth-first search. For an implementation of an ATN parser in LISP one can refer to Gazdar and Mellish [7].

Nondeterminism in ATNs occurs in two places. The first is the choice of an arc to follow out of the current state and the other is the choice of a word sense in the lexicon entry in case of a *category* arc. The first one is solved by the parser which uses depth-first search strategy. The second one is handled by a LISP function library that manipulates the structures returned from the morphological analyzer and the local registers used in the networks. The output of the parser is a tree structure whose nodes keep the contents of the local registers. This structure is manipulated by special functions that are called in the actions part of arcs.

```
(setq networks
 '(

   (S
     ((Registers (subject object
                  predicate ... ))

      (Initial (0)
         (t)
         (
         ))

      (Final   (6)
         (
          ... CONDITIONS ...
         )
         (
          (senprint res "SENTENCE")
         ))

      ;; Subject is a nominative NP
      (From 1 to 2 by NP
         (
          ... CONDITION 2 ...
          (equal (npcase annp) 'NOM)
         )
         (
          ... ACTIONS ...
          (setq subject star)
         )
         (
          ... CONDITION 1 ...
         ))
     )
   )

   (NP
     ((Registers (case agree possess
                  def res ...))

      (Initial (0)
         (t)
         (
         ))

      (Final   (2)
         (
          t
         )
         (
          (npprint res '("  "))
         ))

      ; Pronoun
      (From 0 to 2 by PN
         (
          ... CONDITION ...
         )
         (
          ... ACTIONS ...
           (setq def t)
           (setq case (fcase star))
           (setq possess (fposs star))
           (setq agree (fagree star))
           (setq res (np-struct star))
         ))

     )
   )
 )
)
```

Figure 5.2. Definition of Networks

The network definitions are kept as LISP association lists in a global variable named *networks*. The association list keeps networks separately and within each network its local registers, initial states, final states and arcs have different sections.

Figure 5.2 is a small and simplified portion of the definition of networks. In the figure there are LISP function calls in addition to the LISP built-in predicates. These functions are from the LISP function library that are implemented. These functions save space in the network definitions and makes the networks understandable. The functions in the library can be grouped into four according to their functions:

1. Functions that produce the structures that are manipulated within networks are written for each different network.
2. Functions that extract a feature value from a structure.
3. Functions that pretty-print the output of networks which are essentially the structures that are built.
4. Functions that apply a set of tests on a structure. Once these functions are written, the grammar writer can call these functions with different tests from *conditions* part of the arcs.

The arc types described in the chapter for ATNs are implemented in our parser. It is possible to make the parser allow new arc types but these are sufficient for our implementation. In fact we did not spend much effort on the development of the parser. We concentrated on the design of the grammar instead.

5.1.1 Table Look-up for NP Network

Our ATN parser operate top down, making implicit expectations of what will be found next in the sentence, based on what has been found. Each arc represents an expectation. If an arc is followed, and later the parse fails, the parser backs up to the last choice point and tries an alternative choice. The problem caused by this backup is that certain phrases may be parsed over and over again, each

time yielding the same structure and repeatedly being rejected because the larger structure that it is embedded in does not meet the parser's expectations.

To avoid this problem, we have implemented a modified version of ATN parser which can do table look-up for NP network. We can think of this table look-up operation as a variant of *chart parsing* techniques to be applied on ATN parsers.

A *chart* is basically a data structure in which the parser records its successful attempts to parse subconstituents of the string of words. Once the parser has recorded the presence of a constituent in one part of the string, it never needs to look for the same kind of constituent there again. This represents a significant improvement on the backtracking algorithms used in most ATN systems. The ability of the chart to record, in addition the current goals of the parser leads to the possibility of implementing very sophisticated algorithms [23].

Active chart parsing technique can easily be modified to handle recursive transition networks. In trying to apply the same technique to ATNs, one encounters difficulties in sharing the previously parsed structures. Therefore, we implemented a variant of chart parsing and used charts only for the Noun Phrase (NP) network. Since NP is the most frequently used network that is called from all other networks, it is a good decision to use chart parsing only for NP.

The mechanism works as follows. When an input sentence is read, the parser first finds all possible parses of NPs in that sentence and stores them in a global variable called ***np-table***. For the example *benim evim*, the parser stores the parses of *benim*, *evim* and *benim evim*. This is the first phase of the parse. Once these NPs are found, the parsing of the sentence starts. When a push arc with label NP is reached, the parser does table look-up from the ***np-table***. This parsing procedure keeps track of the place where the NP is called by using a second table (called ***active***) which can keep the place of the place of NP call uniquely. This uniqueness is guaranteed by keeping all the parent networks (with their destination nodes) of the current arc together with the first word from the input tape that caused to the application of this arc.

```
{NP (*CASE* NOM) (*AGR* 3SG) (*POSS* 1SG) (*DEF* T)
    (*WORD* benim evim)
   {*MODIFIER*
      {NP (*CASE* GEN) (*AGR* 1SG) (*POSS* NIL) (*DEF* T)
          (*WORD* benim)
         ((*CAT* PN)
          (*R* ben))
      }
   }
   {*MODIFIED*
      {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* 1SG) (*DEF* T)
          (*WORD* evim)
         ((*CAT* N)
          (*R* ev))
      }
   }
}
```

Figure 5.3. Output of Parser for NP *benim evim (my house)*

5.1.2 Output Structure of the Parser

As mentioned in the previous sections, there are LISP functions for both building the parse structures and printing these structures in a nice format. The structure-building functions differ from network to network according to local registers defined for each network. In fact the structures only keep the contents of the local registers. An example parse output for the NP network for input *benim evim (my house)* can be seen in Figure 5.3.

The figure shows the contents of the local registers of the NP network together with the output of the morphological analyzer for the words. The curly brackets are inserted for making the output more understandable.

5.2 The Sentence Network

The Sentence Network (S) is shown in Figure 5.4 and the constituents handled by the S network's arcs are shown in Figure 5.5.

We can list the factors that influenced the design of S network as follows:

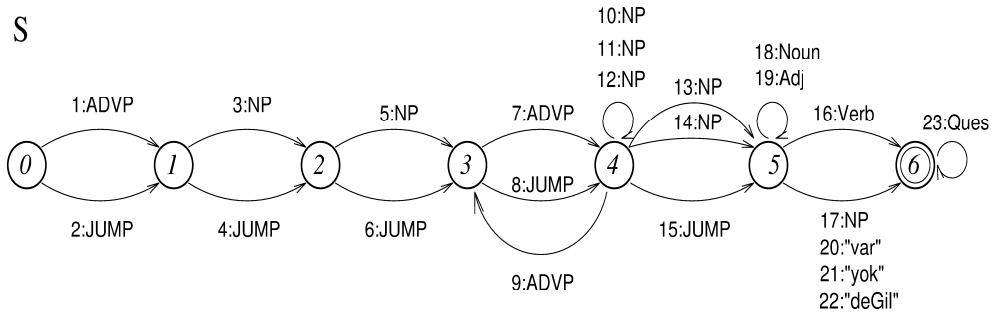

Figure 5.4. Sentence (S) Network

- Since predicate is the only obligatory element in the sentence, it is not bypassed by a JUMP arc. The predicates of verbal and nonverbal sentences are both accepted in parallel arcs at the same location. Other constituents can all be bypassed by JUMP arcs, because they are optional.

- The predicate is put as the last constituent of the sentence, because in Turkish sentence formation rules, the secondary constituents come first and primary constituent which is the predicate comes after them as mentioned in the previous section. The organization of other constituents is chosen to cover the frequently used sentence structures in Turkish [5] that falls in our scope.

- Case information is the most important feature to distinguish subject, object and indirect adjuncts from each other. This can be seen in Figure 5.5 where case information is the most important feature checked as a condition. Indirect objects and indirect adjuncts can not be distinguished until the predicate is reached, because argument structure check can only be done depending on the predicate.

- A nominative object is accepted immediately before the predicate, whereas other constituents are allowed between the accusative object and predicate.

  Nominative object arc has a second function to accept subject in nonverbal and passive-verbal sentences which can not take object. In nonverbal sentences, subject precedes predicate and in passive-verbal sentences the subject is a *supposed subject* which is parsed immediately before the predicate.

  The decision of whether the parsed constituent is a subject or nominative direct object depends on the argument structure of the predicate.

```
Registers:

Subject, Object, Predicate,
LocNP, DatNP, AblNP,
Advadj1, Advadj2, Advadj3,
Res (Result),
CompVerbArg, Question

Conditions and Actions:

S-1: ADVP- A Temporal Adverbial
           Adjunct
     C: Type(*) is TEMPORAL
     A: Set Advadj1 to *.

S-3: NP- Subject
     C: Case(*) is NOMINATIVE
     A: Set Subject to *.

S-5: NP- Accusative Object
     C: Case(*) is ACCUSATIVE
     A: Set Object to *.

S-7: ADVP- Adverbial Adjunct
     A: Set Advadj2 to *.

S-9: ADVP- Adverbial Adjunct
     A: Set Advadj3 to *.

S-10:NP- Locative NP
     C: Case(*) is LOCATIVE
     A: Set LocNP to *.

S-11:NP- Dative NP
     C: Case(*) is DATIVE
     A: Set DatNP to *.

S-12:NP- Ablative NP
     C: Case(*) is ABLATIVE
     A: Set AblNP to *.

S-13:NP- Accusative Object
     C: Case(*) is ACCUSATIVE
     A: Set Object to *.

S-14:NP- Nominative Object
     C: Case(*) is NOMINATIVE
     A: Set Object to *.

S-16:Verb- Verb
     C: Sentence validity check
     A: if CompVerbArg
           Set Predicate to
              (list CompVerbArg *)
        else
           Set Predicate to *
        endif.

S-17:NP- Nominal Predicate
     C: Case(*) is NOMINATIVE &
        CompVerbarg is nil &
        Type(*) is Nonverbal &
        Sentence validity check
     A: Set Predicate to *.

S-18:Noun, S-19:Adj
     C: Case(*) is NOMINATIVE
     A: Set CompVerbArg to *.

S-20:"var", S-21:"yok"
     C: Case(*) is NOMINATIVE &
        CompVerbarg is nil &
        Type(*) is Nonverbal &
        Sentence validity check
     A: Set Predicate to *.

S-22:"deGil"
     C: Case(*) is NOMINATIVE &
        CompVerbarg <> nil &
        Type(*) is Nonverbal &
        Sentence validity check
     A: Set Predicate to
           (list CompVerbArg *).

S-23:Ques
     A: Set Question to *.
```

Figure 5.5. Simplified Conditions and Actions for the S Network

- When the predicate is reached, the validity of the sentence, which is composed of the other constituents parsed so far, is determined according to predicate. Different actions are performed according to the type of predicate. If it is nonverbal then the nominative object, if there is one, is considered as subject. If it is passive-verbal then the nominative object is considered as *supposed subject.*

  Transitivity and causativity are also checked here. In Turkish causative suffix converts an intransitive verb to a transitive verb. If the verb is intransitive without any causative suffix, then it can not take object. If the verb is transitive or intransitive with causative suffix, it can take an object.

- The minimal Turkish sentence requires a predicate with a covert subject. The number of constituents is maximum when the verb is transitive and active. With intransitive and passive verbs, the kinds of constituents used are very limited. In sentences with nominal verbs there is no object and the usage of dative and ablative adjuncts are very rare.

- In Turkish the particles *var (it exists)* and *yok (it does not exist)* are used as predicates. *Var* and *yok* are used with the same endings for other nonverbal predicates as nouns when used as nonverbal predicates. Examples to these cases are: *Odada sandalye var. (There is a chair in the room.), Biz resimde yokuz. (We are not in the picture.).*

- The negative of nonverbal sentences is made with the word *değil (not).* *Değil* is the negative of the verb "to be" and it follows the predicate as a separate word, and personal endings are attached to it. For example, the negation of the nonverbal sentence *Bu kitap benimdir. (This book is mine.)* is obtained as *Bu kitap benim değildir. (This book is not mine.)* using the word *değil.*

The occurrence of constituents in a sentence is determined according to the properties of verb. This feature along with the fact that verb is the last element of a Turkish sentence in frequent usage, makes the ATN parser waste time searching for the unfruitful paths. For example, for an intransitive verb which can not take object, the parser can misinterpret a constituent as object and can not resolve this misinterpretation until the verb is reached.

Since the order of the constituents of a Turkish sentence is very flexible, it

is very difficult to add every kind of sentence structure to an ATN grammar. Instead, the most frequently used orderings have been implemented in S.

The types of sentences that falls into the scope of S network are the frequently used simple sentence structures of Turkish. Handling of complex sentences is done within the NP and ADVP networks. We do not care in S network, whether the structure returned from the NP network is a participle or simply a noun. Here are some important design criteria of the network:

1. Subject (Arc S-3) is generally found in the beginning of the sentence, however in some cases it can be preceded by a temporal adverbial adjunct (Arc S-1).

2. Accusative object is parsed in two different arcs (Arcs S-5 and S-13). This is because the accusative object allow other constituents to come in between itself and predicate. Additional restrictions are put to reject sentences where there is both a nominative and an accusative object.

3. Dative, locative and ablative NPs are parsed together in arcs S-10, S-11 and S-12, because their order within themselves is very flexible and they can be used interchangeably. They are also put closer to the predicate because they can also function as indirect objects. The order of adverbial adjuncts and these NPs is also very flexible and this flexibility is manipulated in the network with the use of a backward arc between states 3 and 4. Some of the structures that can be accepted are: *Dative NP + ADVP + Locative NP*, ADVP + ADVP + Locative NP + Dative NP, Locative NP + Ablative NP + ADVP + ADVP,etc.

4. The arc S-17 is for handling nonverbal sentences. For an NP to function as a predicate of a nonverbal sentence, it should contain a special flag in its structure denoting that usage. NP structures which do not contain this flag can not be used as predicates and similarly NPs which contain this flag can not be used as other constituents like objects, adjuncts,etc. Arcs S-20 and S-21 are put for parsing other nonverbal sentences with *var* and *yok* as predicates. The arc S-22 is for *değil* and it is used in combination with arcs S-18 and S-19.

5. Many verbs in Turkish are compounds, formed by a noun (or adjective) followed by one of the auxiliary verbs *etmek (to do), eylemek (to make), olmak (to be), kılmak (to perform)*. These compounds are very frequent in

Turkish and are included in our grammar. Examples to these compound verbs are: *mutlu olmak (to be happy)*, *telefon etmek (to telephone)*, and *dikkat etmek (to pay attention)*. These compound verbs are parsed using the arcs S-18 and S-19 combined with arc S-16.

5.2.1 Testing the validity of the sentence

As we have mentioned before, the validity of a sentence is determined according to the properties and argument structure information of the verb. With a top-down parsing algorithm for ATNs, we can only do these checks at the end of the sentence (assuming verb is the last element of a Turkish sentence). We have implemented LISP functions, which take the objects that are parsed up to that time together with the argument structure information of the verb and decide whether the parse is valid or not. In these functions, we consider type of the predicate (verbal or nonverbal) and the passive and causative suffixes of the predicate (if it is verbal).

For nonverbal predicates, the functions first check that there is no direct object in the sentence. However, when there is a nominative object that is parsed, the parser first looks at the Subject (whether it is empty or not), and if it is empty nominative object is converted to a subject.

For verbal predicates, first the subject is tested. If the subject is empty and there is a nominative direct object and the predicate can not take a direct object (The verb should be passive without any causative suffix), subject and direct object is exchanged as in the case with nonverbal sentences. After this step arguments of the verbs are tested one by one and finally the causative suffix is tested. Since the causative suffix increase the transitivity of the verb, that is it makes an intransitive verb to take an accusative object and a transitive verb to take a dative object, it should be explicitly handled. Finally, we check that there are no two objects with the same role and all the obligatory objects that the verb should take are taken.

Subject-predicate agreement is also checked here. The rule is as follows: If the predicate's agreement is third person (singular or plural), the subject's agreement should also be third person (singular or plural) without any check of the number. In the other cases the agreements of subject and predicate should totally match both in person and number features.

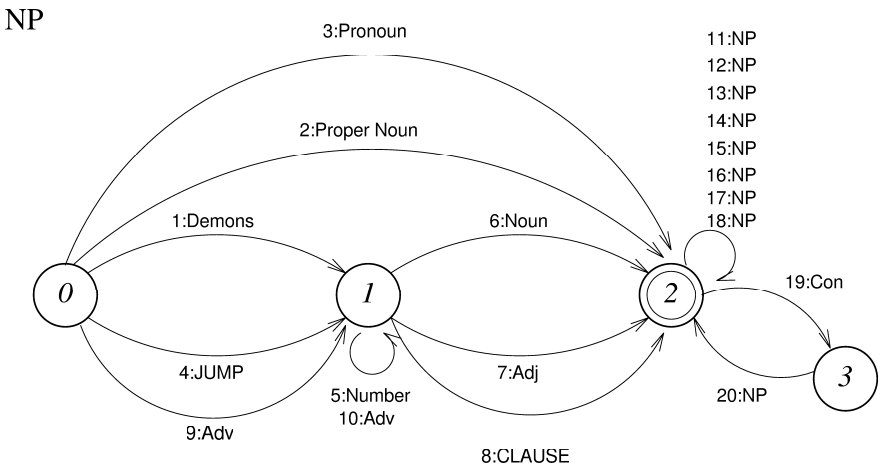

Figure 5.6. Noun Phrase (NP) Network

5.3 The Noun Phrase Network

The S network makes use of the Noun Phrase (NP) network shown Figure 5.6 very frequently. The current implementation of NP, includes nominal compounds and adjectival compounds. Participle and infinitive clauses are also added to the NP network because these clauses can play different syntactic roles within NP. The NP network is used very frequently because an NP can be a subject, an accusative object, a nominative object or an indirect adjunct according to case information.

In the simplest case, a Turkish NP can be a possessive or non-possessive noun, a possessive or non-possessive adjective, a proper noun, a pronoun a participle clause or an infinitive clause. *Bahçe (garden), bahçesi (his/her/its garden), kırmızı (red), yeşili (his/her/its green), Orhan (a proper noun), sen (2nd person singular pronoun)* are examples of these basic NP structures. Nouns and adjectives can take a demonstrator, or suffixless number constructions before them, but proper nouns and pronouns can not. The noun phrases *bu ev (this house), iki elma (two apples)* are also accepted as basic NPs without recursive calls. Adverbs expressing uncountable quantities like *az (few), çok (much)*, comparative adverb *daha (more)* and superlative adverb *en (most)* can be used to modify adjectives within an NP. The structures formed using this rule can be used as modifiers within NP. For example the adverb-adjective combination *en güzel (most beautiful)* can be used to modify the noun *çocuk* to produce the compound NP *en güzel çocuk (most beautiful child).* Therefore, we added these adverbs as separate arcs to the NP network to include these

cases to our grammar. The conditions and actions of the NP network for these cases can be seen in Figure 5.7. [1]

The NP network produces compound NP structures from these simple NPs by recursive calls to itself. These calls enable the network to produce all syntactically correct parses with all ambiguities. The conditions and actions on these *push* arcs state the restrictions of how two NPs can be combined to produce a larger NP structure. The restrictions on these arcs are written such that they satisfy the syntactic properties of the noun phrase to produce the correct parse. They also resolve the conflicts that may occur between the arcs and cause to the production of incorrect or multiple parses (identical parses). These push arcs are listed in Figure 5.8.

The following sections describe the types of compounds that are handled by the current version of NP network with the *push* arcs that they are included in.

5.3.1 Nominal Compounds

Turkish nominal compounds generally consist of two or more nouns. The third person possessive suffix links one noun to another to form nominal compounds. They are classified into definite and indefinite nominal compounds [2, 8, 5, 21].

5.3.1.1 Definite Nominal Compounds

We can further classify definite nominal compounds into genitive-possessive compounds and possessive-compounds because they have different syntactic properties.

1. **Genitive-Possessive Constructions**

   In Turkish, a noun phrase with genitive suffix can modify a noun with possessive suffix to form a genitive-possessive construction. These constructions are also called as *definite nominal compounds* in Turkish syntax. We can give examples as *benim evim (my house), çocuğun kitabı*

[1] The uppercase letters in the figures stand for Turkish letters ç,ğ,ı,ö,ş and ü.

```
Registers:

Agr (Agreement),
Poss (Possessive),
Def (Definite),
Case, Res (Result),
Modifier, Modified,
Demonstrator, Describers,
Adverb, Conjunct

Conditions and Actions:

NP-1:Demons
     A: Set Definite to True.
        Set Demonstrator to *.

NP-2:Proper-Noun
     A: Set Definite to True.
        Set Case to case of *.
        Set Poss to possessive of *.
        Set Agr  to agreement of *.
        Set Res to *.

NP-3:Pronoun
     A: Set Definite to True.
        Set Case to case of *.
        Set Poss to possessive of *.
        Set Agr  to agreement of *.
        Set Res to *.

NP-4:JUMP

NP-5:Number
     C: * is Nominative and
        Non-Possessive
     A: Append * to Describers.

NP-6:Noun
     A: Set Case to case of *.
        Set Poss to possessive of *.
        if Poss <> nil
           Set Definite to True
        endif.
        Set Agr  to agreement of *.
        Set Res to *.

NP-7:Adj
     A: Set Case to case of *.
        Set Poss to possessive of *.
        if Poss <> nil
           Set Definite to True
        endif.
        Set Agr  to agreement of *.
        Set Res to *.

NP-8:CLAUSE
     A: Set Case to case of *.
        Set Poss to possessive of *.
        if Poss <> nil
           Set Definite to True
        endif.
        Set Agr  to agreement of *.
        Set Res to *.

NP-9:Adv
     C: Subcat(*) is COMPARATIVE or
        Subcat(*) is SUPERLATIVE or
        Subcat(*) is QTY-U
     A: Set Adverb to *.

NP-10:Adv
     C: Subcat(*) is COMPARATIVE and
        Subcat(Adverb) is QTY-U
     A: Set Adverb (list Adverb *).
```

Figure 5.7. Conditions and Actions for NP

```
; e.g. benim evim, evin odasI
NP-11:NP
     C: Case is Genitive &
        ...
     A: Set Definite to True.
        ... Details in Figure 5.9

; e.g. kitap kapaGI,
       CalISma masam
NP-12:NP
     C: Case is Nominative &
        ...
     A: Set Definite to True.
        ... Details in Figure 5.10

; e.g. Celik kapI, kitap kapaGI
NP-13:NP
     C: Case is Nominative &
        ...
     A: Set Definite to nil.
        ... Details in Figure 5.11

; e.g. odadaki masa,
       masadaki kalem
NP-14:NP
     C: Case is Nominative &
        Res end with "+de+ki"
        ...
     A: Set Definite to True.
        ... Details in Figure 5.12

; e.g. sabahki adam, dUnkU olaylar
NP-15:NP
     C: Case is Nominative &
        Res end with "+ki"
        ...
     A: Set Definite to True.
        ... Details in Figure 5.13

; e.g. SapkalI kIz, evsiz insanlar
NP-16:NP
     C: Case is Nominative &
        Res end with "+li" or "+siz"
        ...
     A: Set Definite to nil.
        ... Details in Figure 5.14

; e.g. Cikolata renginde ev,
       tahtadan masa
NP-17:NP
     C: Case is Locative or Ablative &
        ...
     A: Set Definite to nil.
        ... Details in Figure 5.15

; e.g. evi gUzel, boyu uzun
NP-18:NP
     C: Case is Nominative &
        Poss is 3PS or 3PL &
        ...
     A: Set Definite to True.
        ... Details in Figure 5.16

NP-19:Con
     A: Set Conjunct to *.

NP-20:NP
     C: if Case(Res) <> Nominative
           Case(Res) = Case(*)
     A: Set Definite to Definite(*).
        Set Res to (list Res Conjunct *).
```

Figure 5.8. Conditions and Actions for NP (continued)

```
NP-11:NP
  C: Case is Genitive &
     if Agr is 3PL or 3PS
         possessive(*) = 3PL or 3PS
      else
         Agr = possessive(*)
     &
     if Definite(*) = True {
        if Modifier(*) exists
          Case(Modifier(*)) =
               Nominative
        else
                                          Possessive(*) exists
                                       }

                                    A: Set Definite to True.
                                       Set Case to case of *.
                                       Set Poss to possessive of *.
                                       Set Agr  to agreement of *.
                                       Set Modifier to Res.
                                       Set Modified to *.
                                       Set Res to
                                           (list Modifier Modified).
```

Figure 5.9. Conditions and Actions for Genitive-Possessive Compounds

(the child's book). These cases are handled in composite NP arc NP-11 in Figure 5.9. [2]

The genitive suffix indicates that the noun to which it is attached is the possessor of some other noun. The possessive suffix indicates that the noun to which it is attached is possessed by some other noun. The genitive and possessive suffixes refer to one another, and both are necessary. However, if the possessor is a pronoun, it can be omitted and the resultant noun phrase is still valid with a hidden pronoun. The noun phrases *benim odam (my room)* and *odam (my room)* both have the same meaning, but their places of acceptance are different. The first of them is handled as a genitive-possessive construction but the second one is handled as a basic NP without any recursive call.

A restriction on such compounds is that the agreement of the modifier must be equal to the possessive of the modified with a small exception in 3rd person cases. Other restrictions that are in Figure 5.9 are for rejecting incorrect parses that this arc can produce.

[2]The conditions on *push* arcs check whether two NPs (NP1 and NP2) can be combined to form a longer NP. (Informally they check whether NP1 + NP2 is still a valid noun phrase.) Contents of NP1 is kept in Res (Result) register. Other local registers Agr, Poss, Modifier, etc,... keep the features for NP1 only. NP2 is kept in global register *, since it is the structure returned from the recursive call to the NP network. Note that these figures may contain LISP notations like *nil (empty)*, because these conditions are all stated as LISP statements in the actual implementation.

Role(Res) is a feature returned from the word lexicon, and it denotes nouns with a *ROLE* feature like measurement, adjective, property, material, etc,...

```
NP-12:NP
  C: Case is Nominative &
     Res does not end with suffixes
       "+ki","+li","+lik",
             "+ci", or "+siz"
     &
     Modifier does not end with
       suffixes "+ki","+li",
         "+lik", "+ci", or "+siz"
     &
     demonstrator(*) is nil &
     if describers(*) exists {
        describers(*) = "bir" or
        Res is an Adjective or
        Res is a Participle or
     }
     else True
     endif.
     &
     if Agr = 3PL
        Poss(*) exists
     else True
     endif.
     &
     if Def = True {
        Demonstrator <> nil or
        Res is a participle or
        Res is an infinitive or
        ( Res is a Proper Noun &
          possessive(*) exists )
     }
     else {
        possessive(*) exists or
        ( * is a Proper Noun &
          Modifier is nil &
          modifier(*) is nil )
     }
     endif.
     &
     if definite(*) {
        ( modifier(*) is nil &
          possessive(*) exists &
            * is not a Proper-Noun )
        or
        ( * is a Proper Noun &
          possessive(*) is nil &
          ( Res is Adjective or
            Role(Res) is Adjective ))
     }
     else True
     endif.
     &
     if possessive(*) is nil {
        ( Res is possessive participle or
          Res is possessive infinitive or
          * is a Proper Noun or
          ( Demonstrator <> nil &
            Poss is nil &
            ( Res is Adjective or
              Role(Res) is Adjective or
              Role(Res) is Material )
            )
        )
     }
     else {
       if modifier(*) exists {
         case of modifier(*) is
                  Nominative &
         modifier(*) does not end with
           suffixes "+ki","+li","+lik",
                    "+ci", or "+siz"
       }
       else True
       endif.
     }
     endif.
  A: Set Definite to True.
     Set Case to case of *.
     Set Poss to possessive of *.
     Set Agr  to agreement of *.
     Set Modifier to Res.
     Set Modified to *.
     Set Res to (list Modifier Modified).
```

Figure 5.10. Conditions and Actions for the arc NP-12

```
NP-13:NP
  C: Case is Nominative &
     Def is nil &
     Res does not end with suffixes
      "+ki","+li","+lik",
            "+ci", or "+siz"
     &
     Modifier does not end with
       suffixes "+ki","+li",
         "+lik", "+ci", or "+siz"
     &
     demonstrator(*) is nil &
     if describers(*) exists {
        describers(*) = "bir" or
        Res is an Adjective or
        Res is a Participle or
     }
     else True
     endif.
     &
     if Agr = 3PL
        Poss(*) exists
     else True
     endif.
     &
     if definite(*) {
        ( modifier(*) is nil &
          possessive(*) exists &
          * is not a Proper-Noun )
     }
     else {
        if modifier(*) exists
           case(modifier(*)) =
                Nominative
           else True
        endif.
     }
     endif.
     &
     if possessive(*) is nil {
        ( Res is non-possessive
                    participle or
          Res is non-possessive
                    infinitive or
          Res is Adjective or
          Role(Res) is Adjective or
          Role(Res) is Material or
          ( Role(Res) is Measure &
            describers <> nil)
        )
     }
     else {
        ( Res is not a participle &
          Res is not an infinitive &
          possessive(*) is 3SG or 3PL &
          Res is not an adjective )
     }
     endif.

  A: Set Definite to True.
     Set Case to case of *.
     Set Poss to possessive of *.
     Set Agr  to agreement of *.
     Set Modifier to Res.
     Set Modified to *.
     Set Res to
         (list Modifier Modified).
```

Figure 5.11. Conditions and Actions for the arc NP-13

```
NP-14:NP
  C: Case is Nominative &
     Res is Locative followed by
         "+ki" suffix
     &
     if Definite(*) = True {
        * is a Proper Noun  or
        Determiner(*) exists  or
        if modifier(*) = nil
           possessive(*) exists
        else {
           case(modifier(*)) =
                Nominative  &
           possessive(*) exists
                                        }
                                     }
                                     else True
                                     endif.

                                A: Set Definite to True.
                                   Set Case to case of *.
                                   Set Poss to possessive of *.
                                   Set Agr  to agreement of *.
                                   Set Modifier to Res.
                                   Set Modified to *.
                                   Set Res to
                                       (list Modifier Modified).
```

Figure 5.12. Conditions and Actions for the arc NP-14

The members of a genitive-possessive construction behave independently. Both of them can take modifiers.

2. **Possessive Compounds**

   Other definite nominal compounds that do not fit to the syntax of the genitive-possessive constructions are handled in arc NP-12 in Figure 5.10. Examples that are accepted by this arc are definite parses of *çocuk kitabı (children's book)*, *at arabası (horse cart)* (with a possible hidden possessor *onun (his/her/its)*), *evim (my house)* (with a hidden possessor *benim (my)*), and *bu çelik kapı (this steel door)*, *Ankara kalesi (Ankara castle)* where the modifier is definite.

   The difference of these compounds from genitive-possessive constructions is that they can only be modified as a whole, whereas in the latter case, both members can be modified by article "bir", adjectives, numerals or demonstrators as mentioned before.

5.3.1.2 Indefinite Nominal Compounds

Indefinite nominal compounds are handled in arc NP-13 in Figure 5.11. In the indefinite compounds like *çelik kapı (steel door)* and *altın bilezik (golden bracelet)*, where the first and second members of the compound are non-possessive, there are limited number of nouns that can be used as the first

```
NP-15:NP
  C: Case is Nominative &
     Res ends with "+ki" suffix
        but without a Locative before
     &
     if Definite(*) = True {
        * is a Proper Noun  or
        Determiner(*) exists  or
        if modifier(*) = nil
           possessive(*) exists
        else {
           case(modifier(*)) =
                Nominative  &
           possessive(*) exists
                                        }
                                     }
                                     else True
                                     endif.

                                  A: Set Definite to True.
                                     Set Case to case of *.
                                     Set Poss to possessive of *.
                                     Set Agr  to agreement of *.
                                     Set Modifier to Res.
                                     Set Modified to *.
                                     Set Res to
                                         (list Modifier Modified).
```

Figure 5.13. Conditions and Actions for the arc NP-15

member. In this kind of compounds the first member defines what the second member is made of or what the second member is like. The indefinite parses of compounds *çocuk kitabı (children's book)*, *at arabası (horse cart)* where a possessor is not present are handled in this arc. The second element of these compounds contains the suffix "possessive suffix third person" and function as a *Compound Marker (CM)* [22]. These compounds behave in the same way as their definite counterparts when they are modified with article "bir", adjectives, numerals and demonstrators.

5.3.2 Adjectival Compounds

An adjective modifying a noun, precedes the noun to form an adjectival compound. Simple adjectival compounds that are composed of two words are produced according to this rule. In the example *kırmızı kalem (red pencil)* the modifier is a nominative adjective and the modified is a noun. The main word in an adjectival compound is the noun. Adjective is used as a secondary element to modify and strengthen the meaning of noun. The simple adjectival compounds have similar syntactic properties with nominal compounds and hence handled together with them in arcs NP-12 and NP-13 in Figures 5.10 and 5.11 to increase the efficiency of grammar.

For other adjectival compounds which show different syntactic properties

```
NP-16:NP
  C: Case is Nominative &
     Def is nil &
     Res ends with one of the suffixes
         "+li","+lik","+ci", or "+siz"
     &
     if Definite(*) = True {
        * is a Proper Noun  or
        Determiner(*) exists  or
        if modifier(*) = nil
           possessive(*) exists
        else {
           case(modifier(*)) =
                Nominative  &
           possessive(*) exists
        }
     }
     else True
     endif.

  A: Set Definite to True.
     Set Case to case of *.
     Set Poss to possessive of *.
     Set Agr  to agreement of *.
     Set Modifier to Res.
     Set Modified to *.
     Set Res to
         (list Modifier Modified).
```

Figure 5.14. Conditions and Actions for the arc NP-16

arcs NP-14 through NP-18 are used. These can be classified as follows:

1. **The Relative Suffix "+ki"**

   The relative suffix "+ki" can occur following the locative case, the genitive case or the nominative case with certain words indicating time [11]. In any syntactic analysis, it is necessary to differentiate the relative suffix after genitive case from the other two because it can only function as a noun, and never as a modifier within a noun phrase by itself. Therefore, we treated the other two as special cases in different places of our grammar.

   The relative suffix "+ki" attached to nouns, pronouns or noun phrases in locative case to form a new structure showing location that can function as an adjective in a noun phrase. Examples to this case are *bahçedeki ağaç (the tree in the garden), bendeki kitap (the book I have)* or as a more complex structure *evin bahçesindeki meşe ağacı (the oak tree in the garden of the house)* where the complete NP *evin bahçesi (garden of the house)* is used as a modifier showing location. These structures employ special properties and hence handled in a separate arc. The conditions and actions are listed in Figure 5.12.

   The relative suffix "+ki" can also be attached to certain nouns indicating

```
NP-17:NP
  C: Case is Ablative or Locative
     &
     ( Res is Adjective or
       Role(Res) is Adjective or
       Role(Res) is Material )
     &
     if Definite(*) = True {
        * is a Proper Noun  or
        Determiner(*) exists  or
        if modifier(*) = nil
           possessive(*) exists
        else {
           case(modifier(*)) =
                Nominative  &
                possessive(*) exists
           }
        }
        else True
        endif.

  A: Set Definite to True.
     Set Case to case of *.
     Set Poss to possessive of *.
     Set Agr  to agreement of *.
     Set Modifier to Res.
     Set Modified to *.
     Set Res to
         (list Modifier Modified).
```

Figure 5.15. Conditions and Actions for the arc NP-17

time like *dün (yesterday), bugün (today), sabah (morning)* and the resultant adjective can function as a temporal modifier in a noun phrase. Examples to this case are *akşamki yağmur (the rain in the evening), dünkü toplantı (the meeting yesterday).* Their usage is more restricted then the previous usage of relative suffix so they are handled in a different arc NP-15 which can be seen in Figure 5.13.

2. **Structures formed by "+li", "+siz" suffixes** [3]

   More adjectival compounds can be produced in Turkish with the attachment of suffixes "+li", "+siz" to the modifier. When attached to a noun these suffixes convert the noun to an adjective. Since they are produced by conversion they are handled in another arc (NP-16) in Figure 5.14. We can give examples as *evsiz insanlar (homeless people), ümitli insanlar (hopeful people).*

   Furthermore, the suffixes "+li" and "+siz" can be attached to adjectival compounds and these compounds can function as an adjective to form longer compounds as in the example *kırmızı başlıklı kız (the girl with the red cap).*

3. **Locative and Ablative Usage in NP**: Locative and ablative NPs can be used as modifiers of other NPs. The usage of ablative within NP is limited to the cases where it denotes the material from which something

[3] "+li" stands for "+lI" and "+siz" stands for "+sIz".

```
NP-18:NP
  C: Case is Nominative &
     Poss = 3PL or 3PS &
     Modifier is nil &
     demonstrator(*) is nil &
     describers(*) is nil &
     modifier(*) is nil &
     ( * is Adjective or
       Role(*) is Adjective or
       Role(*) is Material )

  A: Set Definite to True.
     Set Case to case of *.
     Set Poss to possessive of *.
     Set Agr  to agreement of *.
     Set Modifier to Res.
     Set Modified to *.
     Set Res to
         (list Modifier Modified).
```

Figure 5.16. Conditions and Actions for the arc NP-18

is made. Similarly, the usage of locative is limited to the cases where it has a property feature. To handle these cases we put these features as roles *Material* or *Property* to the root word lexicon. These structures are similar to adjectival compounds but because of case information they have to be handled in another arc. The conditions and actions for arc NP-17 is in Figure 5.15. Some examples are: *çikolata renginde bir yaprak (a leaf with color of chocolate)* and *tahtadan bir masa (a wooden table).*

4. **Adjectives following Possessive Nouns**: In Turkish, adjectives can follow nouns with third person possessive suffix to produce adjectival compounds. These compounds function the same as adjectives and can be used to modify other nouns within NP. In example *bahçesi büyük ev (the house with a big garden)*, the noun *ev* is modified by an adjectival compound *bahçesi büyük* which is an example of these compounds. These structures are handled in arc NP-18 in Figure 5.16.

5.3.3 Structures formed by "+lik", "+ci" suffixes

The suffixes "+lik" and "+ci"[4] can be attached to nouns or adjectives to form nouns or adjectives that can be used as modifiers in compound NPs. They show similar properties with suffixes "+li" and "+siz" and hence handled together with them in arc NP-16 in Figure 5.14. We can give examples as *kunduracı adam (the shoemaker man), kurbanlık koyun (the sheep destined for sacrifice).*

[4] "+lik" stands for "+lIk" and "+ci" stands for "+cI".

One difference of these suffixes from "+li" and "+siz" is that they can not be attached to compounds to form longer compounds.

5.3.4 Chaining of Nominal and Adjectival Compounds

The recursive nature of the NP network also allows chaining of compounds to produce larger compounds. The nominal compound *lokantanın bahçesinin kapısı (the door of the restaurant's garden)* is an example to chaining using the genitive-possessive construction rules. There is no limitation on the size of this chaining and it can be extended as required unless they do not violate the compound NP formation rules. The following examples are taken from Lewis [10] to show some allowable patterns for chaining and their differences. *Ford aile arabası (the Ford family-car), Ford ailesi arabası (the Ford-family car), Ford ailesinin arabası (the car of the Ford family), Ford'un ailesinin arabası (the car of Ford's family)*, and *Ford'un aile arabası (Ford's family car).*

5.3.5 Conjunctions in NP

Conjunctions that can be used to join noun phrases are put to the NP network. These conjunctions are *ve (and), veya (or)* and *ile (with).* In addition to them the punctuation symbol comma can also join two noun phrases to form a bigger NP. These are all included in our grammar by using a new state. The arcs NP-19 and NP-20 are included for this purpose.

Some conditions should be satisfied in arc NP-21 after popping from the NP network. These are mainly on case of the NPs that are planned to be combined. If the case of the NP before conjunction is nominative, then the second NP can be of any case. The cases of the two NPs should be the same if the first NP is not nominative.

5.3.6 Demonstratives, Numerals and the article "bir" in NP

Turkish has three demonstrative pronouns: *bu (this), şu (that)*, and *o (that)*. They indicate the location of an object with respect to the speaker. Demonstrators *şu* and *o* both translated as *that* to English, but in general *o* is used for distant objects and *şu* is used for closer objects.

Demonstratives are handled carefully in our grammar to avoid any wrong parses. Since our NP network proceeds recursively to produce compound NPs, these demonstratives can be at wrong places. For example the two valid NPs *çelik (steel)* and *bu kapı (this door)* can not be combined to form the NP *çelik bu kapı*. These cases are carefully eliminated by special conditions.

Numerals and the article "bir" also require special attention. When an adjective modifies a noun, it precedes that noun. If there is an article "bir" before the noun, the adjective also precedes "bir" as in example *eski ağaç (old tree), eski bir ağaç (an old tree)*. In fact the article "bir" has a special usage in this example. It can also come before the adjective. However in that case its function is different, because it describes the number feature in that case. Examples are : *bir eski ağaç (one old tree)*. The example *on iki eski ağaç (twelve old trees)* is also valid. However numerals can not come in between two words unless the compound is a genitive-possessive compound.

Numerals combined with measurement units (or counting words) like *metre (meter), tane (grain), adet (item), kamyon (truck)* have some special properties. These counting words alone can not be used as a modifier in a noun phrase. But, when they follow a numeral they can produce a compound NP. The construction *bir bardak su (a glass of water)* is valid and slightly different than the possessive compound *su bardağı (waterglass)*.

5.3.7 Participle and Infinitive usage in NP

The CLAUSE network which accepts participles and infinitives is called from the NP network because of its syntactic functions in Turkish NP. In this section we will first describe the syntactic functions of participles within NP followed by a description of infinitives. The structure of the CLAUSE network will be

described later in this chapter.

Because of the multiple functions of participles and infinitives, it was necessary to call the CLAUSE network from the NP network. Once the participle (or infinitive) clause is parsed within NP, it is converted to an NP structure and from that time on, it can be used wherever an NP is used unless it is not eliminated by a special condition.

1. **Participles:** Constructions with participles in Turkish correspond to constructions with relative clauses in English [21]. A participle construction, like an adjective, is part of the noun phrase. The noun that it modifies may be used in any grammatical function in the main sentence; this grammatical function has no effect on the internal organization of the participle phrase. A participle phrase normally precedes all other modifiers of the noun; a demonstrative, however, may precede the participle phrase only if there are no other modifiers. A participle, like any other adjective, may be used itself in a sentence with no noun following: *O dairede oturanlar her gece içki içer. (The people living in that apartment drink every night.).*

   Participles carry temporal information according to the suffixes that are used to produce them. These suffixes are: "+yEn" (present), "+mIş" (past), "+yEcEk" (future), "+dIk" (past), "+yEsI" (future), "+yIcI" (present), and finally "+Ir/+mEz" (aorist). Examples of usage of these suffixes will be given with the clause network.

   In addition to the usage of participle phrases as modifiers in NP, they can function separately within the sentence. They can be used as a subject, object, indirect adjuncts, or even predicates of sentence with suitable case suffixes. Examples for these different functions are as follows:

   An Adjective in NP:
   evde bekleyen misafirler, (The guests who are waiting at home)
   A Modifier in NP:
   kumsalda yürüyenlerin giysileri, (The clothes of the ones who are walking on the beach)
   Subject:
   Atı alan Üsküdar'ı geçti. (An idiom with meaning that it is too late to do something.)
   Object:

Derste konuşmayana kötü not verilir. (Failing grades are given to the ones who do not participate in class.)

Predicate:

Benim inancım bunun gerekli olmadığıdır. (My belief is that this is not necessary.)

2. **Infinitives (Verbal Nouns):** Verbal noun (Infinitive) constructions in Turkish are devices by which one sentence may be included within another to fill the grammatical role of noun phrase within the main sentence. In the example sentence *Orhan'ın geç kalmasına kızdım. (I got angry at Orhan's being late.)*, the object is *Orhan'ın geç kalmasına* which functions as any noun phrase. It is formed from the simple sentence *Orhan geç kaldı. (Orhan is late.)* by the use of a verbal noun suffix "+mE".

   A verbal noun in Turkish is formed by the attachment of suffixes "+mEk" (denotes pure undefined action), "+mE" (denotes a specific action or result of action), and "+yIş" (denotes the manner of action) to verbal stems. These will further be explained in the section for CLAUSE network with their properties.

   Infinitive clauses like participles have many functions in Turkish syntax except that they can not function as adjectives. The examples to their usages are :

   A Modifier in NP:

   konuşmak niyeti, (the intention to speak)

   Subject:

   Pul biriktirmek en büyük zevkidir. (To collect stamps is his biggest pleasure.)

   Object:

   Ekmek almayı unuttum. (I forgot to buy bread.)

   Predicate:

   En büyük özelliği piyanoyu çok iyi çalmasıdır. (His most important characteristics is his playing the piano very well.)

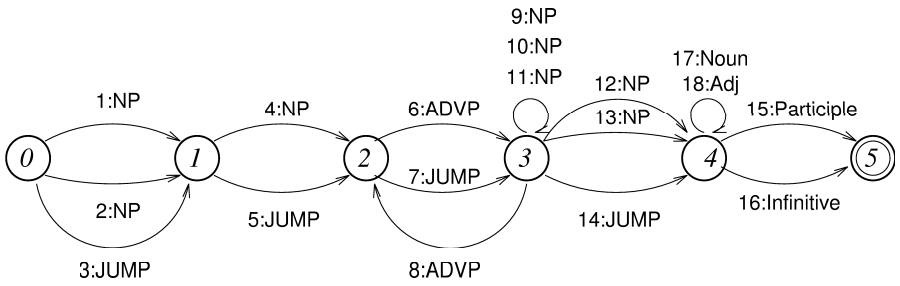

Figure 5.17. CLAUSE Network

```
CLAUSE-1: NP- A Genitive Subject    CLAUSE-11:NP- Ablative NP
CLAUSE-2: NP- A Nominative Subject  CLAUSE-12:NP- Accusative Object
CLAUSE-4: NP- Accusative Object     CLAUSE-13:NP- Nominative Object
CLAUSE-6: ADVP- Adverbial Adjunct   CLAUSE-15:Participle
CLAUSE-8: ADVP- Adverbial Adjunct   CLAUSE-16:Infinitive
CLAUSE-9: NP- Locative NP           CLAUSE-17:Noun (Compound Verbs)
CLAUSE-10:NP- Dative NP             CLAUSE-18:Adj  (Compound Verbs)
```

Figure 5.18. Constituents handled by the arcs of CLAUSE

5.4 CLAUSE Network

Participles and infinitives show similar properties, both according to their syntactic structures and their place of usages. Their place of usage is within the NP network as mentioned before. This choice is preferred because of their multiple functions in the sentence. They can take the functions of a noun phrase and can show up at any place an NP shows up in a sentence. The only difference is that participles can function as adjectives within NP, whereas infinitives can not.

Examples to usages of participles in a Turkish sentence are:

1. *Masada oturan Ahmet, gazetesini okuyor. (Ahmet, who is seated at the table is reading his newspaper.).* The participle suffix used is the present participle "+yEn" which is also called the subject participle.

2. *İstasyona gelmiş olan trenden insanlar iniyordu. (People were getting off the train, which had come into the station.).* The participle suffix used is the past participle "+mIş".

3. *bu sabah çıktığım tepe (the hill that I climbed this morning).* The participle suffix used is the past participle "+dIk" which is also called as the object participle.

4. *Yarın kalkacak vapurla gideceğim. (I will go by the boat that will leave tomorrow.).* The participle suffix used is the future participle "+yEcEk".

5. *şaşılası bir olay (an astonishing event).* The participle suffix used is the future participle "+yEsI".

6. *gezici kütüphane (travelling library).* The participle suffix used is the present participle "+yIcI".

7. *sözünde durur bir erkek (a man who keeps his word).* The participle suffix used is the aorist participle "+Ir".

Examples to usages of infinitives in a Turkish sentence are:

1. *Evlerini bulmak kolay olacak. (It will be easy to find their house.).* The verbal noun suffix used is "+mEk".

2. *Erken yatmaya alışıyorum. (I am getting used to going to bed early.).* The verbal noun suffix used is "+mE".

3. *O adamın çok tuhaf bir yürüyüşü var. (That man has a very strange way of walking.).* The verbal noun suffix used is "+yIş".

The structure of the CLAUSE network is similar to the Sentence (S) and GERUND networks. The placement of adverbial adjuncts and locative, ablative and dative NPs are done in between states 2 and 3 in Figure 5.17 is the same as S network. The placements of accusative and nominative objects is also the same. The validity of the clause is determined when the participle (or infinitive) is reached as in the Sentence (S) network. The main difference of the CLAUSE network is that there is an NP arc (arc CLAUSE-1) for accepting genitive subjects. The participles and infinitives (except infinitives with suffix "+mEk") both can take genitive subjects. In addition to this, participles can take nominative subjects with possessive suffixes, whereas infinitives can not. This small distinction is handled by the network and although it seems possible for an infinitive to take this kind of subject, it is eliminated by extra tests done in the conditions part. The constituents handled by the arcs of the network can be seen in Figure 5.18.

Participles and infinitives behave the same as the verb stem that they are produced from; e.g. a participle produced from a transitive verb has to take a direct object. The verb *beklemek (to wait)* is transitive and its corresponding participle phrase has to take an accusative object as in the example *bizi bekleyenler (the ones who are waiting for us).* The verb *başlamak (to start)* takes a dative as in the example *bu işe başlayanlar (those who are beginning this job).* To incorporate this fact in our grammar, we do checks of the argument structure information of the verb stem that the participle (or infinitive) is produced from. There is however one difference with object participles. Since the object that the verb should take comes after the participle, object participles should be handled differently. As an example to this property, consider the simple sentence *Biz adamı bekledik. (We waited for the man.).* When converted to a relative clause with the object participle "+dik", this sentence becomes *bizim beklediğimiz adam (the man whom we waited)* where the object of the participle, *adam*, follows the participle clause *bizim beklediğimiz.* In such cases, we omitted the check that the verb should take all obligatory objects that it should take.

Another important syntactic property of participles and verbal nouns is that they can take a genitive subject. When the participle (or verbal noun) is followed by a personal suffix, they can take a genitive subject. The participles which can take genitive suffix are "+dIk", "+yEcEk" and "+yEsI" and verbal nouns which can take genitive suffix are "+mE" and "+yIş". The genitive subject can be existent as in the example *kardeşimin beklediği misafir (the guest whom my brother is awaiting for)* or it can be covert as in the example *okuyacağım kitap (the book which I will read).*

The verbal nouns produced by suffix "+mEk" [5] never takes the personal suffixes and hence can not take genitive subjects. In fact these verbal nouns generally do not take subject. Because of this feature, we do not allow any subject in clauses produced by them.

[5]The suffix is termed as the suffix of the infinitive in some grammar books, however in this study we use the terms verbal noun and infinitive interchangeably denoting the words produced by the attachment of suffixes "+mEk","+mE" and "+yIş".

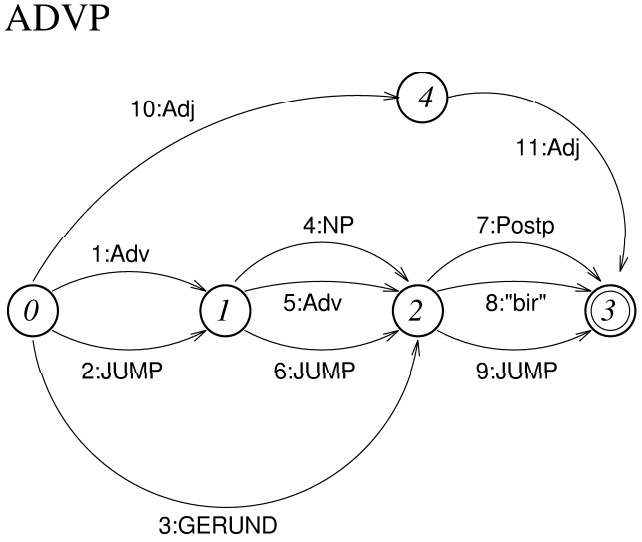

Figure 5.19. Adverbial Phrase (ADVP) Network

5.5 The Adverbial Phrase Network

Adverbial adjuncts modify and strengthen the meaning of verb from mainly time, direction, quantity and quality aspects. There are many different structures used as adverbial adjuncts in Turkish [5]. The structures supported by this work are chosen to cover a frequently used subset of the adverbial adjuncts in Turkish. We did not include adverbial adjuncts which have a very specific syntax in our current work such as *yıllar yılı (years and years), günden güne (from day to day), Allah vere (if only).* We have chosen the ones that can be grouped according to syntactic properties. For example the adverbial adjuncts *ayda bir (once in a month), yılda bir (once in a year)* can be parsed as a temporal NP in locative case followed by "bir", similarly the adverbial adjuncts *dağdan aşağı (down from the mountain), kapıdan içeri (through the door)* can be parsed as an ablative NP followed by one of the postpositions *aşağı, içeri, dışarı, öte, dışarı.* There are many such groupings according to the case of the NP and the type of the postposition. Therefore, the structure *NP + POSTP* is put to the network with arcs ADVP-4 and ADVP-7 in Figures 5.19 and 5.20.

Adverbial adjuncts are generally caseless constituents. However, some temporal noun phrases can function as adverbial adjuncts, although they have case information. Examples are *pazara (sunday), akşamdan (in the evening)* [5].

In addition to the adverbs which can function alone as an adverbial phrase (ADVP), we have common structures like the following :

```
Registers:

Np, Gerund, Postp,
Adv1, Adv2, Adj1,
Res (Result)

Conditions and Actions:

; sabahleyin, iCeri
ADVP-1:Adv
     A: Set Adv1 *.
        Set Res to Adv1.

; okula gelirken, gelince
ADVP-3:GERUND
     A: Set Gerund to *.

; sabaha, iki yIldIr
ADVP-4:NP
     A: Set Np to *.

; Cok Once, daha az
ADVP-5:Adv
     C: (Subcat(*) = TEMP &
         Subcat(Adv1) = TEMP)
        or
        (Subcat(*) = QTY-U &
         Subcat(Adv1) = COMPARATIVE)
     A: Set adv2 *
        Set Res to (list Adv1 Adv2).

; gelinceye kadar, geleli beri
; UC ay Once, daGdan aSaGI
; geldiGimden beri, gelmemden Once
ADVP-7:Postp
     C: Adv1 is nil &
        Adv2 is nil
        if Gerund <> nil {
           .. GERUND + POSTP ..
        }
        else if Np <> nil {
           .. NP + POSTP ..
        }
     A: Set Postp to *.
        if Gerund <> nil {
           Set Res to (list Gerund Postp).
        }
        else if Np <> nil {
           Set Res to (list Np Postp).
        }

; ayda bir, yIlda bir
ADVP-8:"bir"
     C: Adv1 is nil &
        Adv2 is nil &
        Case(Np) is Locative &
        Subcat(Np) is Temporal
     A: Set Postp to *.
        Set Res to (list Np Postp).

ADVP-9:JUMP
     C: if Gerund <> nil {
           .. GERUND ALONE..
        }
        else if Np <> nil {
           .. NP ALONE ..
        }
     A: if Gerund <> nil {
           Set Res to Gerund.
        }
        else if Np <> nil {
           Set Res to Np.
        }

ADVP-10:Adj
     C: Subcat(*) = QUAL or
        Subcat(*) = QTY-C
     A: Set Adj1 to *.

; gUzel gUzel, birer birer
ADVP-11:Adj
     C: * = Adj1,
     A: Set Res to (list Adj1 *).
```

Figure 5.20. Simplified Conditions and Actions for the ADVP Network

1. **Gerunds as adverbial adjuncts:** Gerund clauses alone or followed by a postposition act as adverbial adjuncts. There are many types of gerund clauses. The most commons are ones which have predicates of adverbs. These gerunds are handled in the GERUND network. Examples to a gerund followed by a postposition are: *ölünceye kadar (until he dies)*, *biz buraya geleli beri (since we came here)*. Examples to the usage of gerunds alone can be seen in the Gerund network section. Participle or infinitive clauses also have uses as gerunds with a suitable postposition. These structures are also accepted under the structure **NP + POSTP**, however the NP is checked whether it is a participle or infinitive. Examples to these cases are: *biz otele gelene kadar (until we came to this hotel)*, *ev kirasını ödeyemeyeceğinden başka (apart from the fact that he is not going to be able to pay the rent)*.

2. **Noun phrases as adverbial adjuncts:** The usage of an NP alone as an ADVP is limited to temporal NPs. Postpositions coming after a noun phrase with a suitable case suffix produce ADVPs. For example an ablative NP followed by one of the postpositions *beri (since)*, *önce (before)* or *sonra (after)* produces a temporal adverbial adjunct. A dative NP with temporal feature followed by postposition *kadar (till)* produces a temporal ADVP as in *sabaha kadar (till morning)*, however if the NP has no temporal feature the produced ADVP is directional as in the example *okula kadar (up to school)*.

   Some adverbs can precede noun phrases with temporal feature to produce ADVPs. Examples are *dün sabah (yesterday morning)*, *yarın akşam (tomorrow night)*.

3. **Adverbs and Adjective usage in ADVP:** In addition to the adverbs that can function alone as an adverbial adjunct in sentence, there are structures produced by the combination of two adverbs. Examples to these structures are: *hemen önce (just before)* and *daha çok (more)*.

   Repetition of qualifying adjectives can also function as adverbial adjuncts. When repeated, the adjective *çabuk (quick)* becomes *çabuk çabuk (quickly)* and used as an adverbial adjunct.

GERUND

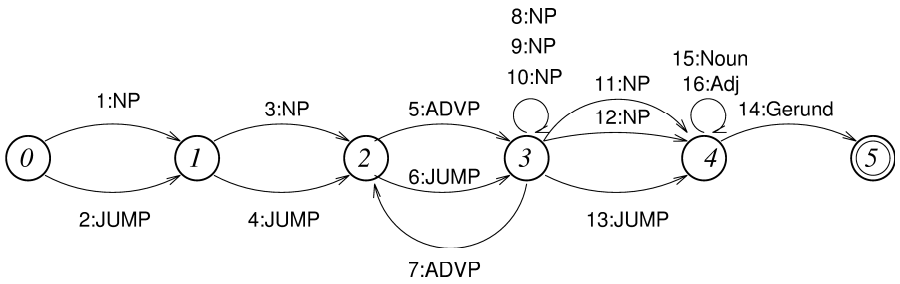

Figure 5.21. GERUND Network

```
GERUND-1: NP- A Nominative Subject  GERUND-10:NP- Ablative NP
GERUND-3: NP- Accusative Object     GERUND-11:NP- Accusative Object
GERUND-5: ADVP- Adverbial Adjunct   GERUND-12:NP- Nominative Object
GERUND-7: ADVP- Adverbial Adjunct   GERUND-14:Gerund
GERUND-8: NP- Locative NP           GERUND-15:Noun (Compound Verbs)
GERUND-9: NP- Dative NP             GERUND-16:Adj  (Compound Verbs)
```

Figure 5.22. Constituents handled by the arcs of GERUND

5.6 Gerund Network

Gerunds are devices by which one sentence may be subordinated to another. In Turkish, this subordination is accomplished by the attachment of certain suffixes to verbal stems. These suffixes are called adverbial suffixes by Underhill [21].

These suffixes can be investigated through the following examples :

1. *Bu akşam kitap okuyup dinlenecektim. (This evening I was going to read a book and rest.).* The suffix that enables subordination is "+yIp".

2. *Küçük kuş ağaçtan düşerek öldü. (The little bird died by falling from the tree.).* The suffix is "+yErEk".

3. *Doktoru görünce ağlamaya başladı. (When he saw the doctor, he began to cry.).* The suffix is "+yIncE".

4. *Evini düşündükçe ağlar. (When he thinks of his home, he weeps.).* The suffix is "+dIkçE".

5. *Arkaya bakmadan devam etti. (He proceeded without looking back.).* The suffix is "+mEdEn".

6. *Ben oradayken öyle kötü bir niyeti yoktu. (While I was there he had no such bad intention.).* The suffix is the adverbial auxiliary "+kEn". The difference of this suffix from the preceding ones is that it is added to predicates and can be added to any type of predicate. Its implementation is the same as preceding cases, because we do not deal with the previous origins of the word in the syntax level.

7. *Koşa koşa içeri girdi. (He came running inside.).* The suffix is "+yE" with duplication of the verb. There are other gerunds that are produced by the duplication of the verb in Turkish syntax like the structure in the example *ben okula gelir gelmez (as soon as I come to school).* These structures are the only structures handled by our grammar. In fact addition of these structures directly to the grammar should be avoided because they put additional overhead to the grammar (We had to add two additional arcs to the GERUND network). What should be done instead is to let a preprocessor to handle these structures and the input to our grammar should be an item denoting that the item can be used alone as a gerund.

The above suffixes are handled in the GERUND network. Since the GERUND network is only called from ADVP network, the result of a call to this network is returned back to the ADVP network where it is converted to an adverbial adjunct of a sentence. The rest of the structures classified as gerunds in books on syntax [8, 5] are postpositional structures produced from the participle (or infinitive) clauses by the addition of a postposition. These cases are omitted from GERUND network and handled elsewhere in the ADVP network as a structure: NP followed by a postposition.

The structure of the GERUND network is similar to the Sentence (S) network. The placement of adverbial adjuncts and locative, ablative and dative NPs are done in between states 2 and 3 in Figure 5.21 is the same as S network. The placements of accusative and nominative objects is also the same. The validity of the gerund clause is determined when the gerund is reached as in the Sentence (S) network. The constituents handled by the arcs of the network can be seen in Figure 5.22.

Chapter 6

Performance Evaluation

This parsing system has been implemented using the Lucid Common LISP programming language in a UNIX^{TM} environment, on SUN SPARC Workstations at Bilkent University.

The size of the our parser is rather small, but the grammar writer is allowed to write LISP functions and integrate them to the *conditions* and *actions* parts of the arcs in order to benefit from all the power of LISP. We have implemented more than 200 LISP functions to increase the power of the system.

In this chapter, we will comment on the results of our parser through examples. We will first discuss left-to-right and right-to-left parsing mechanisms for parsing Turkish sentences. We will later give the example parse trees of the parses that our system is able to find. The original outputs are long and they are put to the appendix.

6.1 Right-to-Left Parsing

The initial design of our parsing system [6] including simple sentence structures of Turkish, did not include complex sentences and it was efficient enough to parse the sentences in its limited scope. When we extended the grammar to cover complex sentences, we faced a left-recursion problem. The reason for this left-recursion was the necessity to call CLAUSE network from the NP network. Since the CLAUSE network also had to call NP to parse its own constituents, a left-recursion occurred.

Our initial solution to this problem was to restrict the number of nested clauses in a noun phrase. That solution was inefficient because of two reasons: First, restricting the number of clauses narrows the scope of our grammar though this may not be a very] important restriction. Second, it is inefficient since the parser has to switch to the CLAUSE network every time the NP network is activated. The same problem occurs between the ADVP and the GERUND networks.

To overcome these limitations, we proposed a *Right-to-Left Parsing* strategy for parsing Turkish complex sentences, and obtained successful results. First, the networks that are discussed in this thesis are reversed to make use of the ATN parser that we were using for *Left-to-Right Parsing.* When this is done, the parser first reverses the input sentence and applies the reversed input on the reversed networks.

The advantage of this method is that it totally eliminates left-recursion and the CLAUSE network is not activated within NP unless a participle or infinitive is seen in the input sentence. One disadvantage of this method is that it decreases the efficiency of the NP network. The compound noun phrase arcs (arcs NP-10..NP-17 in Figure 5.6) causes the problem. For the input *onun evi (his house)*, which becomes *evi onun* when reversed, the parser can not decide which NP arc to follow after the word *evi* is parsed as a simple NP. However, in the original left-to-right parsing scheme the parser had known the arc to follow (the arc NP-10 for genitive-possessive constructions) after the word *onun* was parsed as a simple NP with genitive case. However, compared with the left-recursion problem, this disadvantage is tolerable.

6.2 Parse trees for selected examples

We have selected some input sentences and generated parse trees for them to show the different features of our system. We have used right-to-left parsing scheme and table look-up for NP network that we have explained in section 5.1.1 to obtain these outputs. We will give the parse times as CPU seconds which are taken on a SUN SPARCstation ELC Workstation.

The first example *beyaz tebeşir kutusu (white chalk box)* is a compound noun phrase with two ambiguous parses. The example does not contain an embedded clause and the output is taken rather quickly in approximately 1.7

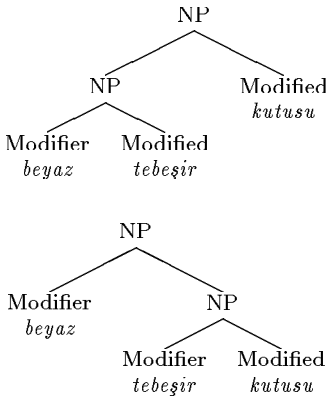

Figure 6.1. Parse trees for NP *beyaz tebeşir kutusu (white chalk box)* (1.7 CPU seconds)

CPU seconds. The parse trees of the example are in Figure 6.1.

The second example *ben okula gelirken ahmet'i gördüm. (I saw ahmet while I was coming to school.)* is a complex sentence with an embedded gerund clause in it. The structural ambiguity in the example is caused by the subject *ben (I)* to be the subject of the main sentence or the gerund clause. There is a lexical ambiguity in the first parse of the sentence caused by the two word senses of the word *ben*, which means *I* as a pronoun and *mole* as a noun. In case of a lexical ambiguity our parser produces different parses that have equivalent parse trees differing at the place lexical ambiguity only. For example, for the *ben* example, the parser produces two parser that have identical parse trees. This is an important factor that increases the number of parses that our system produces, especially in long sentences. The parse trees of the example are in Figure 6.2. The first parse tree in the figure is duplicated because of the lexical ambiguity in the sentence and a total of three parses are produced.

The third example *Ahmet evine dönerken manavdan ne alabileceğini düşünüyordu. (While returning home, ahmet was thinking of what he could buy from the grocery store.)* is a complex sentence with two embedded clauses: a gerund and a participle clause. The parse trees are in Figures 6.3, 6.4 and 6.5. The number of structural ambiguities increase as the number of embedded clauses increase in the sentence. In a complex sentence the constituents that are parsed as noun phrases or adverbial phrases can be joined to the main sentence or any of the embedded clauses unless they do not disobey the syntactic rules of the sentence (or clause). The other structural ambiguity for

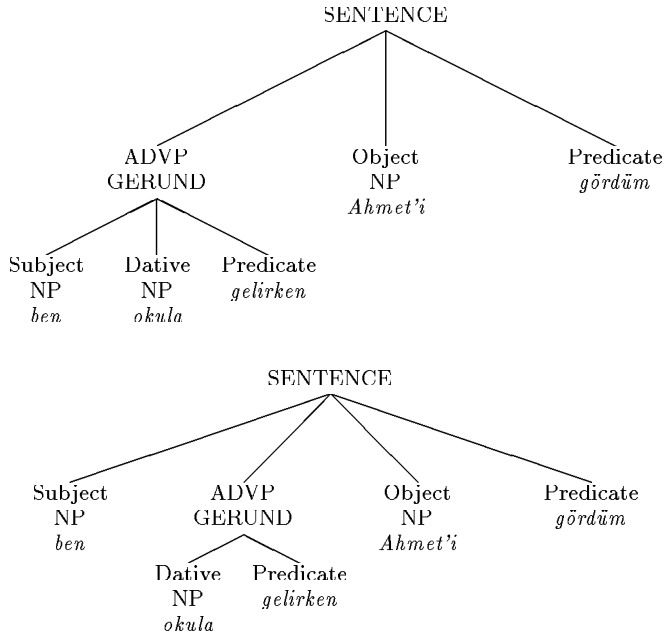

Figure 6.2. Parse trees for S *ben okula gelirken ahmet'i gördüm. (I saw ahmet while I was coming to school.)* (16.0 CPU seconds)

this example is caused by the word *ne (what).* The word *ne* have two lexical ambiguities: as an adverb or as an adjective. Both of the usages of *ne* are syntactically correct for this example. The lexical ambiguities that increase the number of parses that are actually produced by our parser are caused by the lexical ambiguities for the words *evine (to his house)* which have three parses *ev+3SP+DAT*[1], *ev+2SP+DAT*[2], *evin+DAT* and *alabileceğini (he could buy)* which have two parses *alabilecek+3SP+ACC, alabilecek+2SP+ACC.*

The fourth example *dün tanıştığımız genç benim sınıf arkadaşımdı. (The young man that we met yesterday was my school friend.)* is a nonverbal sentence. The parse trees are in Figures 6.6 and 6.7. The parses in Figure 6.6 are produced because of the second word sense of the word *ben* which is a noun and can be modified within a noun phrase.

Our last example in this section *destekleme alımlarının yükünün azaltılması için kurumlar küçültülerek devreden çıkartılmalıdır. (The associations must be closed by diminishing in order to decrease the load of the supporting purchases.)* which is a complex sentence containing two infinitives, one participle and one

[1]3SP stands for third person singular possessive suffix, DAT stands for dative case
[2]2SP stands for second person singular possessive suffix, ACC stands for accusative case

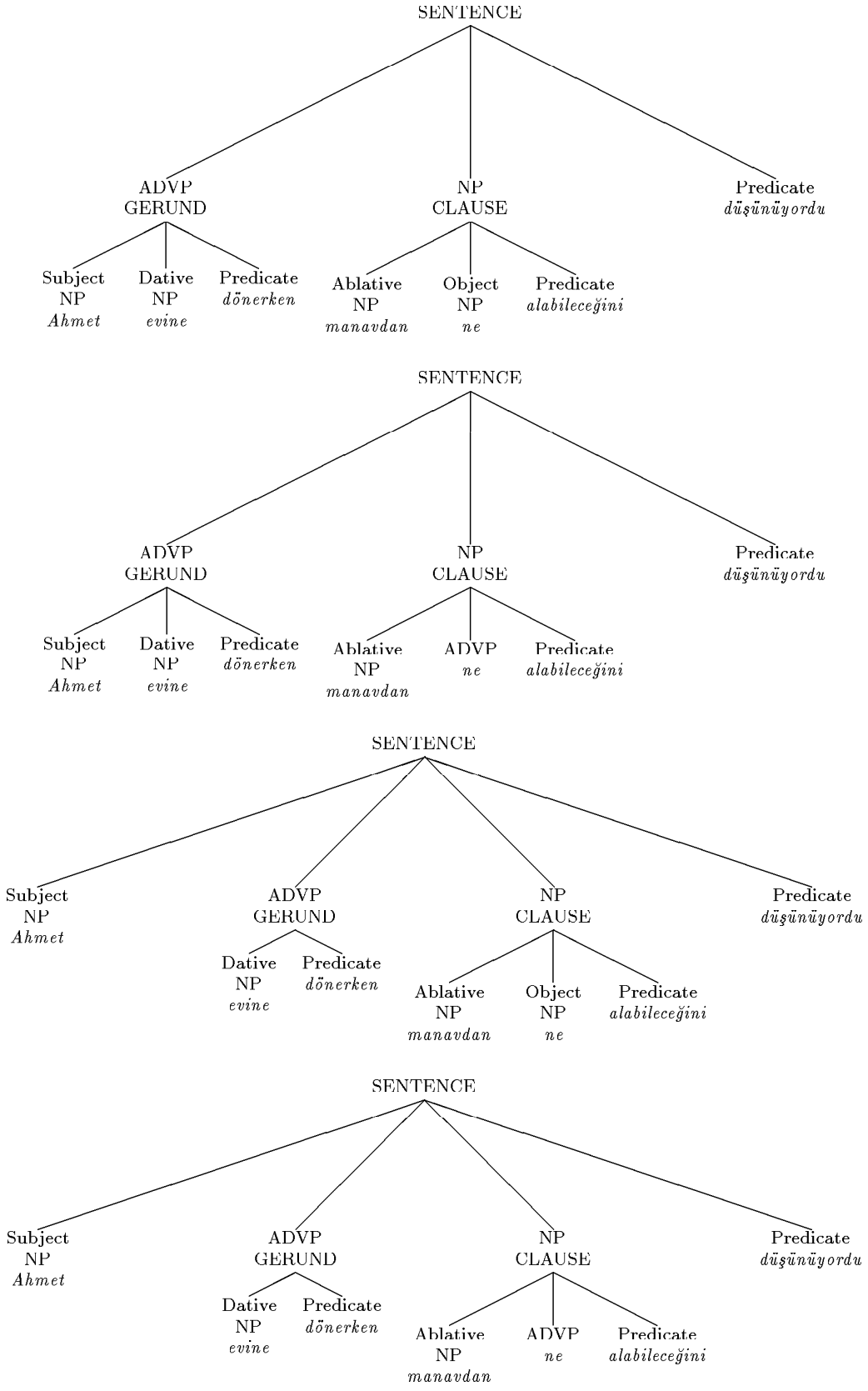

Figure 6.3. Parse trees for S *Ahmet evine dönerken manavdan ne alabileceğini düşünüyordu. (While returning home, Ahmet was thinking of what he could buy from the grocery store.)* (81.0 CPU seconds)

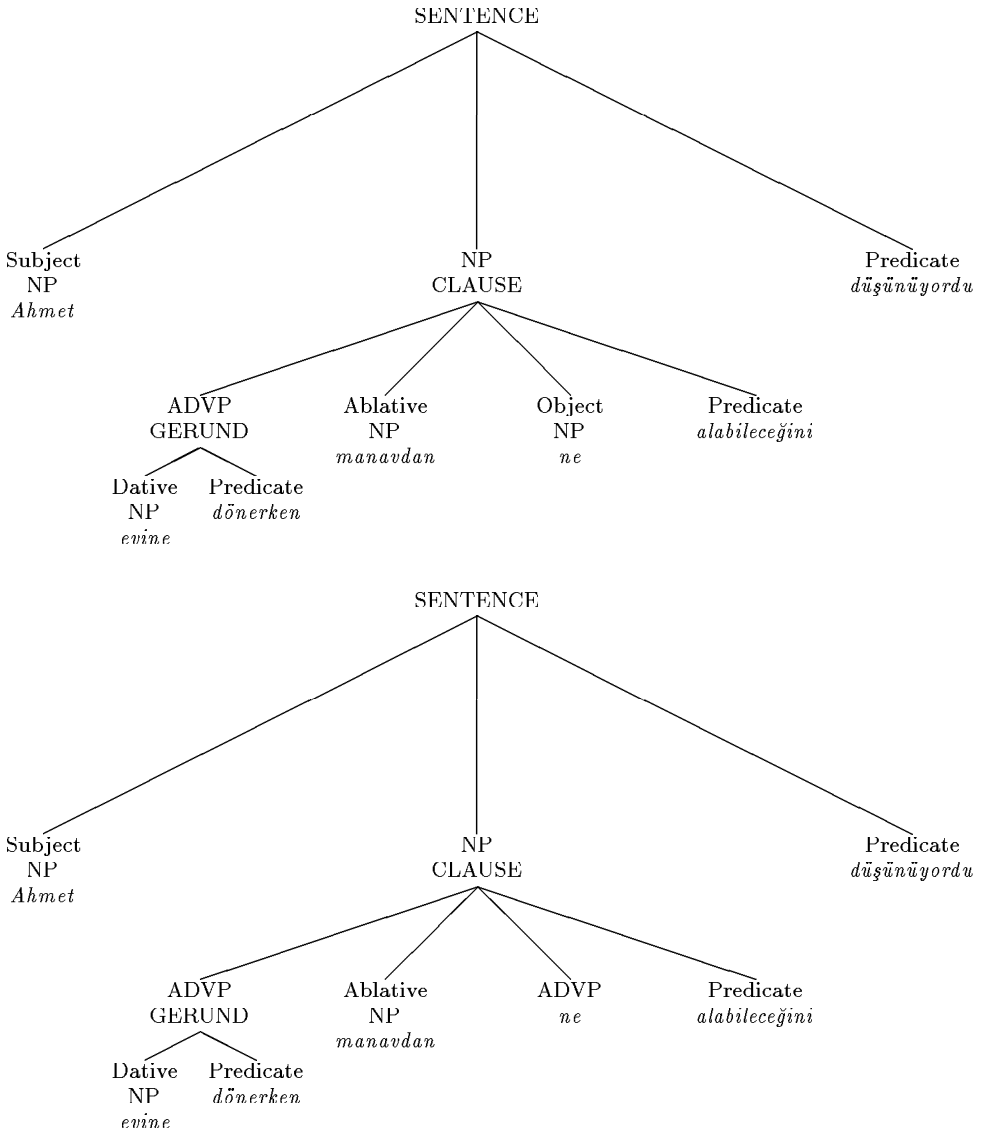

Figure 6.4. Parse trees for S *Ahmet evine dönerken manavdan ne alabileceğini düşünüyordu. (While returning home, Ahmet was thinking of what he could buy from the grocery store.)* (81.0 CPU seconds)

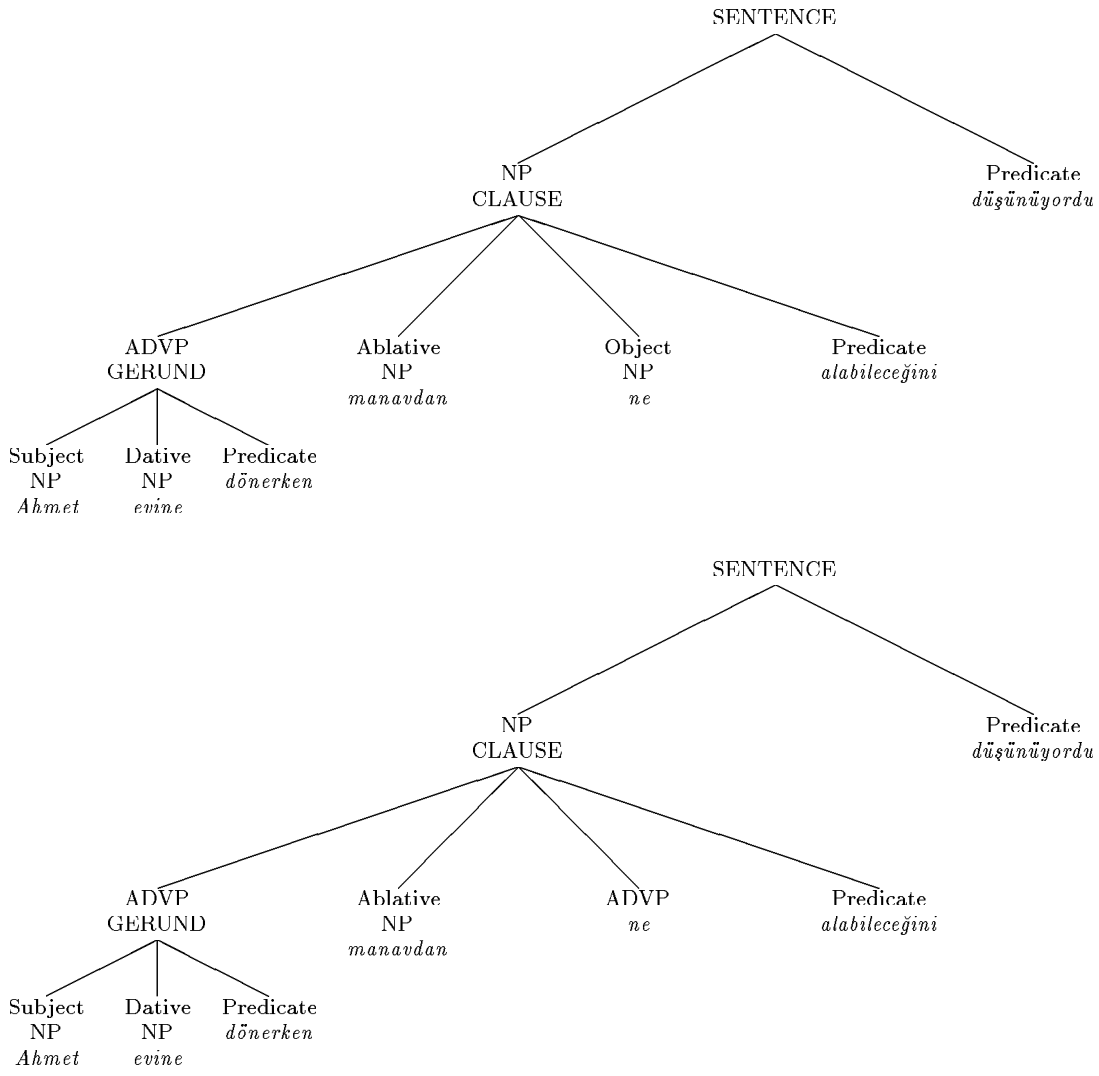

Figure 6.5. Parse trees for S *Ahmet evine dönerken manavdan ne alabileceğini düşünüyordu. (While returning home, Ahmet was thinking of what he could buy from the grocery store.)* (81.0 CPU seconds)

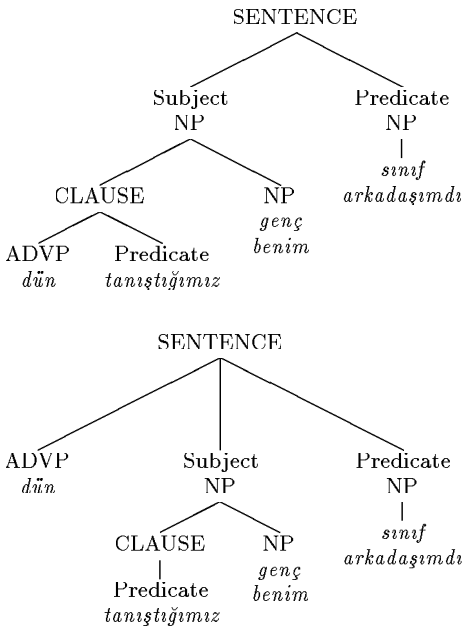

Figure 6.6. Parse trees for S *dün tanıştığımız genç benim sınıf arkadaşımdı. (The young man that we met yesterday was my school friend.)* (80.0 CPU seconds)

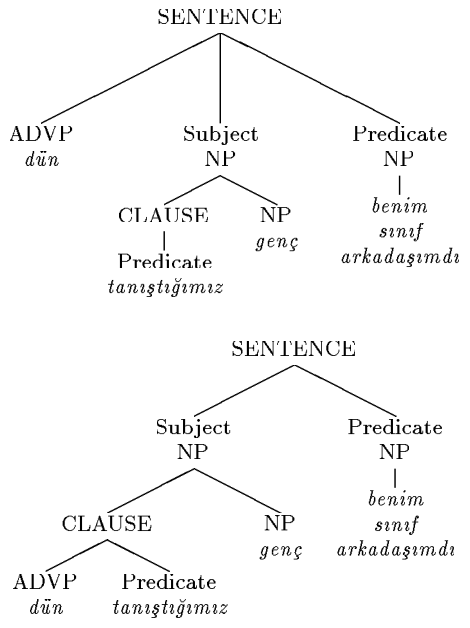

Figure 6.7. Parse trees for S *dün tanıştığımız genç benim sınıf arkadaşımdı. (The young man that we met yesterday was my school friend.)* (80.0 CPU seconds)

	Text 1	Text 2	Text 3	Text 4	Text 5
Total # of Sentences	14	9	9	111	73
Total # of Failures	0	0	1	13	13
Total # of Parses	27	110	104	744	684
Average # of Parses	1.93	12.22	11.56	6.70	9.37
Total # of S Calls	14	9	9	111	73
Average # of S Calls	1	1	1	1	1
Total # of NP Calls	42	23	1157	839	3834
Average # of NP Calls	3.0	2.56	128.56	7.56	52.52
Total # of ADVP Calls	152	340	365	1146	2306
Average # of ADVP Calls	10.86	37.78	40.56	10.32	31.59
Total # of CLAUSE Calls	2	1	62	37	325
Average # of CLAUSE Calls	0.14	0.11	6.89	0.33	4.45
Total # of GERUND Calls	5	49	31	156	346
Average # of GERUND Calls	0.36	5.44	3.44	1.41	4.74
Total CPU Time (sec)	148.9	1622.6	2557.3	2498.9	6835.3
Average CPU Time (sec)	10.6	180.3	284.1	22.5	93.6

Table 6.1. Statistical Results

gerund in it. There are five structurally ambiguous parses of the sentence which are drawn in Figures 6.8 and 6.9. The parse time of the sentence is rather slow because of the embedded clauses in it.

6.3 Statistical Results for Example Texts

We have tested our system on example texts taken from different sources. The first three texts are taken from a book which aims to teach Turkish to foreigners [15] and the last text is taken from a story book for young children. We have made small modifications on the original texts for the compound sentences that are out of our scope. The original texts are given in Appendix A and the texts that we worked on are given in Appendix B.

Table 6.1 lists the statistical results that we have obtained from testing our system on example texts. The first text is parsed quite fast. As seen from the rows for CLAUSE and GERUND networks, the number of embedded clauses in Text 1 is very low. As the number of embedded clauses increase, the parse time increases. This is the reason behind the increase in parse time for Text 2. As we move to Text 3 which is a more complex text than the previous ones, we come across a sentence at which our parser fails. The reason for this failure

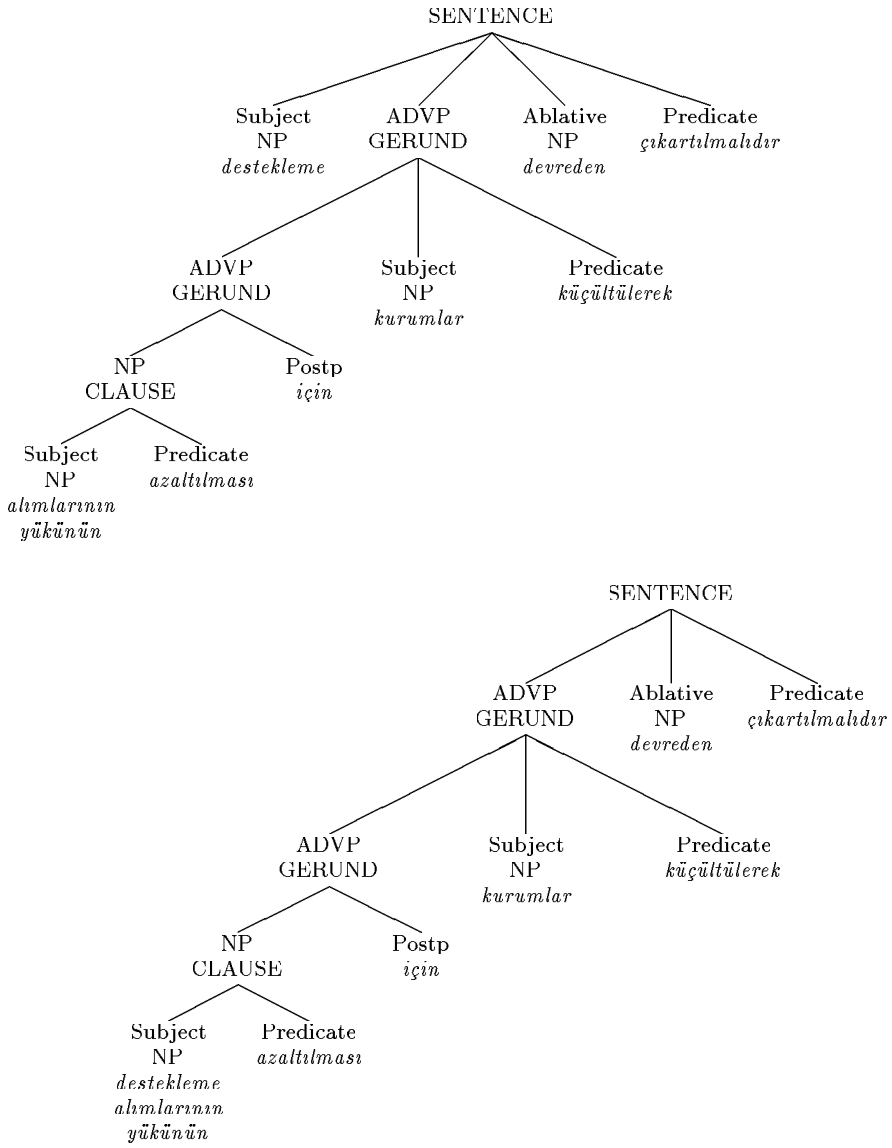

Figure 6.8. Parse trees for S *destekleme alımlarının yükünün azaltılması için kurumlar küçültülerek devreden çıkartılmalıdır. (The associations must be closed by diminishing in order to decrease the load of the supporting purchases.)* (513.55 CPU seconds)

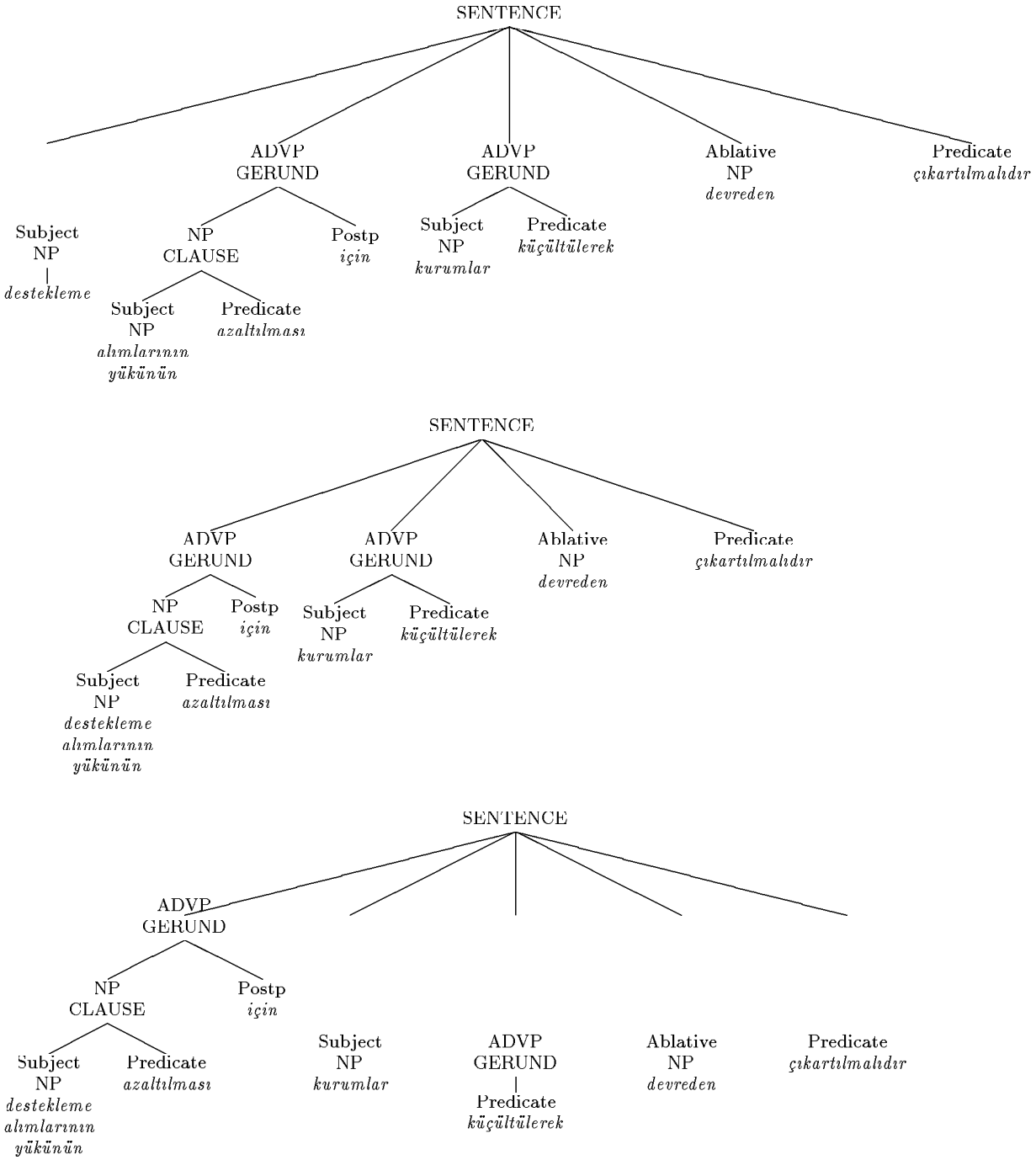

Figure 6.9. Parse trees for S *destekleme alımlarının yükünün azaltılması için kurumlar küçültülerek devreden çıkartılmalıdır. (The associations must be closed by diminishing in order to decrease the load of the supporting purchases.)* (513.55 CPU seconds)

is that the structure of the sentence is out of the scope of our S network. The reason that increases the number of parses for Text 3 is the future tense suffix used throughout the text. Since future tense suffix and the future participle suffix are the same suffix (the suffix "+yEcEk"), the number of parses that are produced increases for Text 3. In addition to the verbal sentence parses of the sentences in future tense, nonverbal sentence parses whose predicates are participle clauses are generated. Text 4 contains 13 unparsed sentences. The failures are due to the limitations that we made when we settled the scope of our grammar. The decrease in average parse time for Text 4 is caused some of the very simple sentences in the text composing of one or two words only. Finally, Text 5 contains 13 unparsed sentences and the failures are because of the sentences that are out of our scope.

To demonstrate the example outputs of our system we have chosen the sentence *kırmızı top yere çarptıkça şarkılar söylüyordu. (The red ball was singing songs as it hit the ground.)*. Four parses are found by our system which are listed in Appendix C. The example demonstrates how a lexical ambiguity (the word *kırmızı* have three ambiguous parses *kırmız+3SP, kırmız+ACC* as nouns and finally *kırmızı* as an adjective) may lead to extra parses. It also demonstrates how an embedded gerund clause may increase the number of parses.

Chapter 7

Conclusions

In this thesis, we have presented an ATN grammar for parsing simple and complex Turkish sentences. This is one of the first efforts in syntactic parsing of Turkish interfaced with a two-level morphological analyzer that uses a large lexicon. This will be an important step of the computational linguistics and natural language studies on Turkish. The results of our system can be used as a first step in various NLP applications such as machine translation, natural language interfaces for databases and computer-aided instruction systems for teaching Turkish language to foreigners. Syntactic analysis is an important step in all of these applications before any further analysis can be performed.

Despite some of its drawbacks, the recursive nature of ATNs provides a powerful structure for developing grammars for natural languages. The parsing system that is developed is fast enough when the sentence contains less number of participle, infinitive and gerund clauses. Since the parser that we use is a top-down parser, the increase in the number of clauses increases the number of networks to be traversed and the search space, which in turn results in a decrease in efficiency. However, even in this case the first parse is found quite successfully, but finding all parses takes time since all possible paths have to be searched one by one.

For a given sentence the parser may produce syntactically correct, but semantically dubious parses. Different meanings can be associated with words or phrases, and most of them result in misleading or useless parses. Some of these misleading parses can be avoided using a system in between the parsing system and the morphological analyzer to eliminate the misleading word senses by looking at the local place of the word in the sentence. This will eliminate

some of the incorrect parses, but a semantic analysis tool should definitely be used to eliminate other incorrect parses.

The major obstacle in front of a syntactic parsing system for Turkish is the word-order freeness of the language. The place of a constituent in a sentence can be changed according to the importance that is given to it within the sentence. This property of Turkish language resulted in many Jump arcs that decreased the power of the system. We have also faced some problems caused by the ATN formalism during the implementation. We have developed a table look-up method for the NP network which is used very frequently to parse the lower-level constituents in other networks to eliminate the duplicate parsing of these constituents. We have also proposed right-to-left parsing of Turkish language using a top-down parsing formalism.

An extension to the grammar that we have developed may be the addition of ordered sentences and compound sentences that contain substantival sentences. However, these kinds of sentences will require recursive calls to the Sentence (S) network within itself, which will result in a decrease in efficiency of the system.

There is an obvious need for automated processing of Turkish text in various applications. Since, computational natural language studies on Turkish are very rare, a lot of work should be done on Turkish.

Bibliography

[1] E. L. Antworth. PC-KIMMO: A two-level processor for morphological analysis. *Summer Institute of Linguistics*, 1990.

[2] T. Banguoğlu. *Turkcenin Grameri (Turkish Grammar).* Türk Dil Kurumu, Ankara, Türkiye, 1986.

[3] D. Bobrow and B. Fraser. An augmented state transition network analysis procedure. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 557–568, 1969.

[4] N. Chomsky. *Aspects of the Theory of Syntax.* M.I.T. Press, Cambridge,Mass., 1965.

[5] R. Şimşek. *Örneklerle Türkçe Sözdizimi, (Turkish Syntax with Examples).* Kuzey Matbaacılık, Trabzon, Türkiye, 1987.

[6] C. Demir and K. Oflazer. An ATN grammar for a subset of turkish. In *Proc. of the 2nd Turkish Artificial and Neural Networks Conference*, İstanbul, Türkiye, 1993. Boğaziçi University.

[7] G. Gazdar and C. Mellish. *Natural Language Processing in LISP.* Addison-Wesley, 1989.

[8] T.N. Gencan. *Dilbilgisi (Turkish Grammar).* Ankara Üniversitesi Basımevi, Ankara, Türkiye, 1979.

[9] R. M. Kaplan. Augmented transition networks as psychological models of sentence comprehension. *Artificial Intelligence*, 3:77–100, 1972.

[10] G. L. Lewis. *Turkish Grammar.* Oxford University Press, 1991.

[11] R. H. Meskill. *A Transformational Analysis of Turkish Syntax.* Mouton, The Hague, Paris, 1970.

[12] H. M. Noble. *C8.9: Natural Language Processing.* Blackwell Scientific Publications, Oxford, 1988.

[13] K. Oflazer. Two-level description of Turkish morphology. In *Proceedings of the sixth Conference of the European Chapter of the Association for Computational Linguistics*, April 1993.

[14] K. Özgüven. An approach to machine translation. In *Proc. of the 1st Turkish Artificial and Neural Networks Conference*, Ankara, Türkiye, 1992. Bilkent University.

[15] Bengisu Rona. *Turkish in Three Months.* Hugo's Language Books Limited, Hertford, England, 1989.

[16] Z. Sagay. A computer translation of English to Turkish. Master's thesis, METU, Ankara, Türkiye, 1981.

[17] S. Shapiro. Generalized augmented transition network grammars for generation from semantic networks. *American Journal of Computational Linguistics*, 8:12–25, 1982.

[18] A. Solak. Design and implementation of a spelling checker for Turkish. Master's thesis, Bilkent University, Ankara, Türkiye, 1991.

[19] H. Tennant. *Natural Language Processing.* Petrocelli Books, 1981.

[20] J. Thorne, P. Bratley, and H. Dewar. The syntactic analysis of English by machine. In D. Michie, editor, *Machine Intelligence.* American-Elsevier, New York, 1968.

[21] R. Underhill. *Türk Dili Grameri (Turkish Grammar).* MIT Press, 1985.

[22] G. van Schaaik. The treatment of Turkish nominal compounds in FG. In M. Fortescue, P. Harder, and L. Kristoffersen, editors, *Layered structure and reference in a functional perspective.* John Benjamins Publishing Co., Amsterdam, 1992.

[23] T. Winograd. *Language as a Cognitive Process, Volume 1: Syntax.* Addison-Wesley, Mass., 1983.

[24] W. A. Woods. Transition network grammars for natural language analysis. *Communications of ACM*, 13:591–602, 1970.

[25] W. A. Woods. Progress in natural language understanding: An application to lunar geology. In *Proc. AFIPS Conference 42*, New York, 1973. AFIPS-Press.

Appendix A

Original Texts

A.1 Text 1

geCen ay marmaris'e gittik.[1] otelimiz denizin kIyIsIndaydI. otelin bUyUk bir bahCesi ve geniS bir plajI vardI. plaj Cok gUzeldi. otelin yemekleri de Cok gUzeldi; garsonlar ve servis Cok iyiydi. her odada banyo ya da duS vardI. bazI akSamlar radyoyu aCtIk ve mUzik dinledik. otelimizin karSIsInda kUCUk bir ada vardI; bir gUn o adaya gittik, orada denize girdik. marmaris'de on beS gUn kaldIk ve Cok gUzel bir tatil yaptIk.

A.2 Text 2

gelecek hafta bir toplantI iCin ankara'ya gideceGim. ankara'da UC gUn kaldIktan sonra uCakla izmir'e geCeceGim. izmir'de bUyUk bir fabrika kuruyoruz. bu fabrika ingiltere'deki fabrikalar gibi olacak. fabrikadaki iSCi sayIsI sekiz yUz elli kadar. bu iSCilerin bazIlarInI UCer aylIk bir sUre iCin ingiltere'ye gOndereceGiz, bOylece biraz ingilizce OGrenecekler. izmir'de fabrikanIn mUdUrU ve diGer kiSilerle konuStuktan sonra londra'ya dOneceGim. gezim hakkInda kIsa bir rapor yazacaGIm.

[1] The uppercase letters in the texts and example parses stand for Turkish letters ç,ğ,ı,ö,ş and ü.

A.3 Text 3

ingiltere ile tUrkiye arasInda tUrkiye'nin baSkenti ankara'da dUn baSlayan gOrUSmeler devam ediyor. Ozellikle ticaret ve turizm konularInIn gOrUSUldUGU toplantIda iki taraf isteklerini ortaya koydu. tUrkiye ingiltere'ye daha Cok mal ihraC etmek istiyor. aynI Sekilde ingiltere de tUrkiye'ye sattIGI mallarda artIS bekliyor. tUrkiye ayrIca yabancI Sirketlerin Ulkede yatIrIm yapmalarInI saGlamaya CalISIyor. turizm yabancI Sirketler iCin Cekici olan bir yatIrIm alanI, ve bir Cok ortak proje Uzerinde CalISIlIyor. tUrkiye'de enflasyonun yUksek olmasI durumu gUCleStiriyor, ancak hUkUmet tarafIndan alInan bazI Onlemlerle bunun dUSmesi bekleniyor.

A.4 Text 4

bir zamanlar kUCUk , parlak , kIrmIzI bir top vardI. o , dUnyadaki toplarIn hepsinden Cok zIplardI. bUtUn toplardan daha hIzlI koSardI. zIplarken , koSarken de gUzel SarkIlar sOylerdi. bu gUzel top kUCUk yIlmaz'In topuydu. gUneSli bir gUnde yIlmaz topunu aldI. bahCenin OnUndeki yola CIktI. topunu yere vura vura yUrUdU. kIrmIzI top yere CarptIkCa: - tInn tIn tIn! diye , SarkIlar sOylUyordu. yIlmaz bir aralIk Cok dUz bir yere geldi. durdu , topunu taSa hIzla CarptI. kIrmIzI top: - tInn tIn! diye SarkIsInI sOyledi. sonra havaya CIktI. bulutlara doGru yUkseldi. yIlmaz ellerini havaya kaldIrInca tekrar geriye dOndU. bu oyun yIlmaz'In Cok hoSuna gitti. sevinCle baGIrdI: - haydi benim gUzel topum! bu sefer daha yUkseklere! topunu bUtUn kuvvetiyle yere CarptI. kIrmIzI top bu sefer: - tInn , tInnn! diye daha uzun bir SarkI sOyledi. yIlmaz ellerini aCtI , bekledi. eyvah! kIrmIzI topa ne oldu? bir tUrlU geri gelmiyordu. yIlmaz elleri aCIk bekledi... bekledi... ama boSuna! kIrmIzI top o kadar Cok sICramIStI ki , bUyUk Cam aGacInIn dallarI arasInda kaybolup gitti. yIlmaz CamIn dallarIna baktI. CalIlarIn , otlarIn arasInI aradI. kIrmIzI topunu bir tUrlU bulamadI. - gUzel topum kayboldu , diye aGlamaya baSladI. sevgili Cocuklar: yIlmaz'In gUzel topu kaybolmamIStI. Citin ObUr tarafIna , otlarIn arasIna dUSmUStU. orada bir o yana , bir bu yana birkaC kere sICradI. sonra tepeden aSaGIya yuvarlanmaya baSladI. minik tavSan onun sesini duydu. merak etti. baSInI yuvasIndan dISarI CIkardI. kIrmIzI top onun burnuna Oyle hIzlI vurdu ki , az kalsIn tavSancIGIn bIyIklarInI koparacaktI. minik tavSan: - aman , bu ne canavar Sey! diye baGIrdI. eliyle burnunu ovalaya

ovalaya yuvasIna sokuldu. korkudan bir daha dISarIya CIkmadI. kUCUk kIrmIzI top gittikCe hIzlandI. kirpi oralarda otlamaya CIkmIStI. az kalsIn ona da CarpacaktI. kUCUk kIrmIzI top Cok korktu. CUnkU kirpinin sivri dikenleri kIrmIzI topa bir batarsa bir daha zIplayamazdI. bereket versin , o sIrada bir taSa CarptI. hIzla sICradI. kirpinin Uzerinden atladI. yuvarlana yuvarlana CayIra indi. orada inek anne ile yavrusu otluyorlardI. inek anne kIrmIzI topa bakmadI bile... CUnkU o CayIrda oynayan Cocuklarda tUrlU tUrlU toplar gOrmUStU. ama kUCUk buzaGI bOyle bir Seyi ilk defa gOrUyordu. kIrmIzI topu gOrUnce telaSlandI: - muu mu! Suradan bUyUk bir elma yuvarlanIyor. onu tutup yiyeceGim , dedi. inek anne: - o elma deGil. yenecek Sey de deGil. senin iSine yaramaz. bIrak yuvarlansIn! dedi. kUCUk buzaGI kIrmIzI topun arkasIndan bir kere daha baktI: bu kIrmIzI elmadan baSka bir Sey olamaz , dedi. topun arkasIndan koSmaya baSladI. kIrmIzI top gittikCe daha hIzlI yuvarlandI. kUCUk buzaGI koStu... koStu... bir tUrlU onu yakalayamadI. kIrmIzI top o kadar hIzlanmIStI ki , artIk gOrUnmez oldu. yIlmaz , topunu Cok aradI. bulamayacaGInI anladI. aGlaya aGlaya eve dOnUyordu. o sIrada CiftCi hasan amca kOpeGi Comar'la gezmeye CIkmIStI. yIlmaz'I gOrUnce: - ne var , neye aGlIyorsun yIlmaz? diye sordu. yIlmaz olanlarI anlattI. hasan amca: aGlama yIlmaz , dedi , kIrmIzI topun belki bizim CayIra yuvarlanmIStIr. gel bir kere de orayI arayalIm. bizim Comar , kaybolan Seyleri bulmakta Cok ustadIr. birlikte CayIra gittiler. ali dayI her tarafI aradI... yIlmaz saGa sola koStu... Comar onlarIn ne aradIklarInI anlayamamIStI. o , tavSan arIyorlar sandI. saGa sola koSuyor , olanca sesiyle havlIyor keskin burnu ile orayI burayI kokluyordu. hasan amca: buralarda yok , biraz daha aSaGIya inelim , dedi. anne inekle buzaGInIn yanIndan geCtiler. kUCUk gOlUn yakInlarIna geldiler. Comar , hasan amca ile gezmeye CIkInca bu gOle girer , orada yIkanmayI Cok severdi. gOle yaklaSInca onlarIn yanIndan ayrIldI. koSa koSa suyun kenarIna geldi. iCine girmedi. kIyIda havlamaya baSladI. Comar acaba neden bOyle telaSlanmIStI , biliyor musunuz? CUnkU kIrmIzI top durgun temiz suyun yUzUnde pIrIl pIrIl parlIyordu. Comar onu gOrmUStU. hasan amca ile yIlmaz'In ne aradIklarInI o zaman anlamIStI. hemen suya atladI. kIrmIzI topu aGzIna aldI. sudan CIkardI. yIlmaz'In ayaklarInIn dibine bIraktI. yIlmaz , Comar'In Islak baSInI okSadI. sevincinden: - yaSa Comar! diye baGIrdI. hasan amcaya teSekkUr etti. eve dOnerken yolda kIrmIzI topunu yere vuruyor , sICratIyordu. ama onu yUkseklere CIkartmamak iCin Cok dikkat ediyordu.

A.5 Text 5

oldukCa sIcak bir gUndU. bir Seye ihtiyacIm olmadIGI halde CarSIda almIStIm soluGu yine. bir saatlik yemek molasInI ancak bOyle deGerlendiriyordum. kalabalIGa aldIrdIGIm bile yoktu. aklIm CalIStIGIm sigorta Sirketindeydi. mesleGimi ve can bey'i dUSUnUyordum durmadan. onu keSfetmeye CalISIyordum. sekreter olarak CalIStIGIm Sirkete sekiz ay Once girmiSti can bey. evli deGildi ama kIzlarIn baSInI dOndUrdUGUne gOre boS sayIlmazdI muhakkak. emin olduGum tek Sey gerCekten cazibeli ve etkileyici bir insan olmasIydI. benim gibi sIradan bir kIza bakacak tip deGildi. bUyUk bir maGazaya girip iCini gezmeye baSladIm. yolum kozmetik reyonuna dUStU. makyajla bir ilgim yoktu pek. arada bir rimel sUrerdim o kadar. zorla gUzellik olmayacaGIna inanmIStIm bir kez. gerCekten bir postere takIldI gOzUm. bir parfUm reklamIydI bu: allure parfume yazIlIydI Uzerinde. altInda da kUCUk harflerle ufak bir not: allure ile cazibenize dayanamayacak. posterdeki kIza tatlI tatlI gUlUmseyen delikanlIyI can bey'e benzettim birden. elimi Ceneme koyup yakISIklI delikanlIyI incelemeye koyuldum. gerCekten benziyor muydu acaba? yoo , yoo her halde delirmeye baSlamIStIm. haftalardIr baktIGIm her yerde onu gOrmeye baSlamIStIm. aslInda posterdeki adam hiC de ona benzemiyordu. arkamdaki ses birden dUSUncelerimi daGIttI. size yardIm edebilir miyim kUCUk hanIm? bir rUyadan uyanIr gibiydim. allure gerCekten etkileyici bir koku. hiC denediniz mi? hayIr anlamIna baSImI salladIm. Oyleyse deneyin bir kez. yoo , buraya parfUm almak iCin gelmedim ben. gOzUm yine postere takIlmIStI. ama bir kez denesem fena olmayacak , aGzImdan CIkan kelimelere dikkat etmemiStim. yanImda o kadar para olup olmadIGInI bile bilmiyordum. gUnlerce biriktirdiGim parayI kasaya sayarken gOzUm hala posterdeydi. evet , evet ona benziyordu! iSe 5 dakika geC kaldIm. hanImlar tuvaletine koStuGumda bizim kIzlar hala sallana sallana makyajlarInI tamamlIyorlardI. tuvaletin birine girip gOzlerimi kapadIktan sonra bileklerime , boynuma aceleyle parfUm sIktIm. aniden etrafImI saran aGIr kokuyla birlikte onunda hayali gOzlerimde canlanmIStI. CIkInca kIzlarIn arasIndan hIzla sIyrIlIp odama koStum. onlar gevezelik ededursunlar , ben her zamanki gibi Onceden masama geCip , can bey yazI getirince onlardan Once kapacaktIm. masama oturur oturmaz can bey elinde bir tomar kaGItla kapIda gOrUndU. milletin dOnmesini bekliyormuS deGer. nazikCe kaGItlarI uzatIrken bunlarI en kIsa zamanda yetiStireceGinden eminim sevgi dedi parfUmUn bUyUsU ces aret vermiSti bana. o gUne kadar yUzUne bakmaya sIkIldIGIm gence doGru

gOzlerimi dikip , hiC kuSkunuz olmasIn dedim. sonra birden kaGItlarI elimden alIp , sana Ozellikle gOstermek istediGim bir paragraf var diye devam etti. o bOlUme Ozen gOsterirsen memnun olurum. kelimeler biraz hatalI yazIlmIS da ... seni SaSIrtmasIn. eGilip baktIm. her Sey gayet dUzgUn ifade edilmiSti. gUlUmsedim. oyalanmak ister gibi bir tavrI vardI masamda. eline kalemi alIp bir iki cUmleyi dUzeltmek isterken parmaklarIma dokundu. elimi hafifCe yana kaydIrIrken gUlUmsUyordum. kIzlar hala dOnmediGi iCin yanImda rahat olduGunu dUSUndUm. birden doGrulup yUzUme baktI. bu akSam CIkISta birlikte bir Seyler iCebilir miyiz? oldukCa tedirgin bir hali vardI. bir derdiniz mi var? iyi gOrUnmUyorsunuz diye sordum. yoo , aksine Cok iyiyim. ancak siz hala bir cevap vermediniz bana. kIzlar masalarIna dOnmeye baSlamISlardI bile.

Appendix B

Pre-edited Texts

B.1 Text 1

geCen ay marmaris'e gittik. otelimiz denizin kIyIsIndaydI. otelin bUyUk bir bahCesi ve geniS bir plajI vardI. plaj Cok gUzeldi. otelin yemekleri de Cok gUzeldi. garsonlar ve servis Cok iyiydi. her odada banyo veya duS vardI. bazI akSamlar radyoyu aCtIk. mUzik dinledik. otelimizin karSIsInda kUCUk bir ada vardI. bir gUn o adaya gittik. orada denize girdik. marmaris'de on beS gUn kaldIk. Cok gUzel bir tatil yaptIk.

B.2 Text 2

gelecek hafta bir toplantI iCin ankara'ya gideceGim. ankara'da UC gUn kaldIktan sonra uCakla izmir'e geCeceGim. izmir'de bUyUk bir fabrika kuruyoruz. bu fabrika ingiltere'deki fabrikalar gibi olacak. fabrikadaki iSCi sayIsI sekiz yUz elli kiSi kadar olacak. bu iSCilerin bir kIsmInI UCer aylIk bir sUre iCin ingiltere'ye gOndereceGiz. bOylece biraz ingilizce OGrenecekler. izmir'de fabrikanIn mUdUrU ve baSka kiSilerle konuStuktan sonra londra'ya dOneceGim. daha sonra kIsa bir rapor yazacaGIm.

B.3 Text 3

ingiltere ile tUrkiye arasInda dUn baSlayan gOrUSmeler ankara'da devam ediyor. Ozellikle ticaret ve turizm konularInIn gOrUSUldUGU toplantIda iki taraf da isteklerini ortaya koydu. tUrkiye ingiltere'ye daha Cok mal ihraC etmek istiyor. ingiltere de tUrkiye'ye sattIGI mallarda artIS bekliyor. tUrkiye ayrIca yabancI Sirketlerin Ulkede yatIrIm yapmalarInI saGlamaya CalISIyor. turizm yabancI Sirketler iCin Cekici olan bir yatIrIm alanIdIr. bir Cok ortak proje Uzerinde CalISIlIyor. tUrkiye'de enflasyonun yUksek olmasI durumu zora sokuyor. hUkUmet tarafIndan alInan bazI Onlemlerle bunun dUSmesi bekleniyor.

B.4 Text 4

bir zamanlar kUCUk , parlak , kIrmIzI bir top vardI. o dUnyadaki toplarIn hepsinden Cok zIplardI. bUtUn toplardan daha hIzlI koSardI. zIplarken koSarken de gUzel SarkIlar sOylerdi. bu gUzel top kUCUk yIlmaz'In topuydu. gUneSli bir gUnde yIlmaz topunu aldI. bahCenin OnUndeki yola CIktI. topunu yere vura vura yUrUdU. kIrmIzI top yere CarptIkCa SarkIlar sOylUyordu. yIlmaz bir aralIk Cok dUz bir yere geldi. durdu. topunu taSa hIzla CarptI. kIrmIzI top SarkIsInI sOyledi. sonra havaya CIktI. bulutlara doGru yUkseldi. yIlmaz ellerini havaya kaldIrInca tekrar geriye dOndU. bu oyun yIlmaz'In Cok hoSuna gitti. sevinCle baGIrdI. haydi benim gUzel topum. bu sefer daha yUkseklere. topunu bUtUn kuvvetiyle yere CarptI. kIrmIzI top bu sefer daha uzun bir SarkI sOyledi. yIlmaz ellerini aCIp bekledi. kIrmIzI topa ne oldu. bir tUrlU geri gelmiyordu. yIlmaz elleri aCIk bekledi. bekledi. ama boSuna. kIrmIzI top o kadar Cok sICramIStI. bUyUk Cam aGacInIn dallarInIn arasInda kaybolup gitti. yIlmaz CamIn dallarIna baktI. CalIlarIn , otlarIn arasInI aradI. kIrmIzI topunu bir tUrlU bulamadI. gUzel topum kayboldu. yIlmaz'In gUzel topu kaybolmamIStI. Citin ObUr tarafIna , otlarIn arasIna dUSmUStU. orada bir o yana , bir bu yana birkaC kere sICradI. sonra tepeden aSaGIya yuvarlanmaya baSladI. minik tavSan onun sesini duydu. merak etti. baSInI yuvasIndan dISarI CIkardI. kIrmIzI top onun burnuna Oyle hIzlI vurdu. az kalsIn tavSancIGIn bIyIklarInI koparacaktI. aman bu ne canavar Sey. eliyle burnunu ovalaya ovalaya yuvasIna sokuldu. korkudan bir daha dISarIya CIkmadI. kUCUk kIrmIzI top gittikCe hIzlandI. kirpi oralarda otlamaya CIkmIStI. az kalsIn ona

da CarpacaktI. kUCUk kIrmIzI top Cok korktu. kirpinin dikenleri kIrmIzI topa batabilirdi. o sIrada bir taSa CarptI. hIzla sICradI. kirpinin Uzerinden atladI. yuvarlana yuvarlana CayIra indi. inek ile yavrusu orada otluyorlardI. inek kIrmIzI topa bakmadI. o CayIrda oynayan Cocuklarda tUrlU tUrlU toplar gOrmUStU. kUCUk buzaGI bOyle bir Seyi ilk defa gOrUyordu. kIrmIzI topu gOrUnce telaSlandI. Suradan bUyUk bir elma yuvarlanIyor. onu tutup yiyeceGim. o elma deGil. yenecek Sey de deGil. senin iSine yaramaz. kUCUk buzaGI kIrmIzI topun arkasIndan bir kere daha baktI. bu kIrmIzI elmadan baSka bir Sey olamaz. topun arkasIndan koSmaya baSladI. kIrmIzI top gittikCe daha hIzlI yuvarlandI. kUCUk buzaGI koStu. koStu. bir tUrlU onu yakalayamadI. kIrmIzI top o kadar hIzlanmIStI. artIk gOrUnmez oldu. yIlmaz topunu Cok aradI. bulamayacaGInI anladI. aGlaya aGlaya eve dOnUyordu. o sIrada CiftCi kOpeGiyle gezmeye CIkmIStI. yIlmaz'I gOrUnce sordu. neye aGlIyorsun. yIlmaz olanlarI anlattI. kIrmIzI topun belki bizim CayIra yuvarlanmIStIr. bir kere de orayI arayalIm. kaybolan Seyleri bulmakta Comar Cok ustadIr. birlikte CayIra gittiler. ali her tarafI aradI. yIlmaz saGa , sola koStu. Comar onlarIn ne aradIklarInI anlayamamIStI. olanca sesiyle havlIyor. keskin burnu ile orayI , burayI kokluyordu. buralarda yok. biraz daha aSaGIya inelim. inekle buzaGInIn yanIndan geCtiler. keskin burnu ile orayI , burayI kokluyordu. kUCUk gOlUn yakInlarIna geldiler. Comar hasan ile gezmeye CIkInca bu gOle girerdi. orada yIkanmayI Cok severdi. gOle yaklaSInca onlarIn yanIndan ayrIldI. koSa koSa suyun kenarIna geldi. iCine girmedi. kIyIda havlamaya baSladI. Comar acaba neden bOyle telaSlanmIStI. CUnkU kIrmIzI top durgun temiz suyun yUzUnde pIrIl pIrIl parlIyordu. Comar onu gOrmUStU. hasan ile yIlmaz'In ne aradIklarInI o zaman anlamIStI. hemen suya atladI. kIrmIzI topu aGzIna aldI. sudan CIkardI. yIlmaz'In ayaklarInIn dibine bIraktI. yIlmaz Comar'In Islak baSInI okSadI. onu yUkseklere CIkartmamak iCin Cok dikkat ediyordu.

B.5 Text 5

oldukCa sIcak bir gUndU. bir Seye ihtiyacIm olmadIGI halde soluGu yine CarSIda almIStIm. bir saatlik yemek molasInI ancak bOyle deGerlendiriyordum. kalabalIGa aldIrdIGIm bile yoktu. aklIm CalIStIGIm sigorta Sirketindeydi. durmadan mesleGimi ve can'I dUSUnUyordum. onu keSfetmeye CalISIyordum. can sekreter olarak CalIStIGIm Sirkete sekiz ay Once girmiSti. evli deGildi ama kIzlarIn baSInI dOndUrdUGUne gOre boS sayIlmazdI. emin

olduGum tek Sey gerCekten cazibeli ve etkileyici bir insan olmasIydI. benim gibi sIradan bir kIza bakacak tip deGildi. bUyUk bir maGazaya girip iCini gezmeye baSladIm. yolum kozmetik reyonuna dUStU. makyajla pek bir ilgim yoktu. arada bir rimel sUrerdim. zorla gUzellik olmayacaGIna bir kez inanmIStIm. gerCekten gOzUm bir postere takIldI. bu bir parfUm reklamIydI. Uzerinde parfUm yazIlIydI. altInda da kUCUk harflerle ufak bir not vardI. bu parfUm ile cazibenize dayanamayacak. birden posterdeki kIza gUlUmseyen delikanlIyI can'a benzettim. elimi Ceneme koyup yakISIklI delikanlIyI inceledim. acaba gerCekten benziyor muydu. hayIr herhalde delirmeye baSlamIStIm. baktIGIm her yerde onu gOrmeye baSlamIStIm. aslInda posterdeki adam ona benzemiyordu. arkamdaki ses birden dUSUncelerimi daGIttI. size yardIm edebilir miyim. bir rUyadan uyanIr gibiydim. bu parfUm gerCekten etkileyici bir kokudur. hiC denediniz mi. hayIr anlamIna baSImI salladIm. Oyleyse deneyin bir kez. hayIr ben buraya parfUm almak iCin gelmedim. gOzUm yine postere takIlmIStI. bir kez deneyeyim. aGzImdan CIkan kelimelere dikkat etmemiStim. yanImda o kadar para olup olmadIGInI bile bilmiyordum. gUnlerce biriktirdiGim parayI kasaya sayarken gOzUm posterdeydi. evet ona benziyordu. iSe beS dakika geC kaldIm. hanImlar tuvaletine koStuGumda kIzlar hala sallana sallana makyajlarInI tamamlIyorlardI. tuvaletin birine girip gOzlerimi kapadIktan sonra bileklerime , boynuma aceleyle parfUm sIktIm. aniden etrafImI saran aGIr kokuyla birlikte onun hayali gOzlerimde canlanmIStI. CIkInca kIzlarIn arasIndan hIzla sIyrIlIp odama koStum. ben her zamanki gibi Onceden masama geCip can yazI getirince onlardan Once kapacaktIm. masama oturur oturmaz can elindeki bir metre kaGItla kapIda gOrUndU. milletin dOnmesini bekliyormuS. bunlarI en kIsa zamanda yetiStireceGinden eminim. parfUmUn bUyUsU bana cesaret vermiSti. o gUne kadar yUzUne bakmaya sIkIldIGIm gence doGru baktIm. hiC kuSkunuz olmasIn. sonra birden kaGItlarI elimden alIp devam etti. sana Ozellikle gOstermek istediGim bir paragraf var. o bOlUme Ozen gOster. kelimeler biraz hatalI yazIlmIS. seni SaSIrtmasIn. eGilip baktIm. her Sey epey gUzel ifade edilmiSti. gUlUmsedim. masamda oyalanmak ister gibi bir tavrI vardI. eline kalemi alIp iki cUmleyi dUzeltmek isterken parmaklarIma dokundu. elimi hafifCe yana kaydIrIrken gUlUmsUyordum. kIzlar henUz dOnmediGi iCin yanImda rahat olduGunu dUSUndUm. birden doGrulup yUzUme baktI. bu akSam CIkISta birlikte bir Seyler iCebilir miyiz. oldukCa tedirgin bir hali vardI. bir derdiniz var mI. iyi gOrUnmUyorsunuz. yoo aksine Cok iyiyim. ancak siz hala bana bir cevap vermediniz. kIzlar masalarIna dOnmeye baSlamISlardI.

Appendix C

Output Example

The sentence *kırmızı top yere çarptıkça şarkılar söylüyordu. (The red ball was singing songs as it hit the ground.)* is parsed and the following outputs are generated. There is a total of 4 parses. The usage of networks for this sentence are: S network 1 time, NP network 3 times, AP 13 times, GERUND 5 times and finally CLAUSE network is never used because there is no participle or infinitive in the sentence. The parse took 10.7 CPU seconds.

```
PARSE 1 :

{ SENTENCE (*SUBJ-TYPE* TRANS-ACT-VERBAL)
 Subject :
   {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* 3SG) (*DEF* T)
       (*WORD* kIrmIzI) (*HPOSS* onun)
      ((*CAT* N)
       (*R* kIrmIz))
   }
 Adverbial Adjunct :
   {ADVP (*TYPE* TEMP)
         { GERUND (*SUBJ-TYPE* ACT-SUB-VERBAL)
          Subject :
            {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
                (*WORD* top) (*HPOSS* NIL)
               ((*CAT* N)
```

```
            (*R* top))
          }
        Dative   :
        DIRECT (*OCC* OPTIONAL) (*ROLE* GOAL)
          {NP (*CASE* DAT) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
            (*WORD* yere) (*HPOSS* NIL)
           ((*CAT* N)
            (*R* yer))
          }
        Predicate :
          {(*TYPE* GERUND) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
            (*WORD* CarptIkCa) (*ROOT* Carp)
           ((*CAT* ADV)
            (*CONV-FROM*
                    ((*CAT* V)
                     (*R* Carp))
                    (*WITH-SUFFIX* dikce))
            (*SUB* TEMP))
          }
       }
  }
Direct Object :
DIRECT (*OCC* OBLIGATORY) (*ROLE* THEME)
  {NP (*CASE* NOM) (*AGR* 3PL) (*POSS* NIL) (*DEF* NIL)
      (*WORD* SarkIlar) (*HPOSS* NIL)
     ((*CAT* N)
      (*R* SarkI))
  }
Predicate :
  {(*TYPE* VERBAL) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
      (*WORD* sOylUyordu) (*ROOT* sOyle)
     ((*CAT* V)
      (*R* sOyle)
      (*ASPECT* PR-CONT)
      (*TENSE* PAST)
      (*AGR* 3SG))
  }
```

```
}

PARSE 2 :

{ SENTENCE (*SUBJ-TYPE* TRANS-ACT-VERBAL)
 Adverbial Adjunct :
   {ADVP (*TYPE* TEMP)
         { GERUND (*SUBJ-TYPE* ACT-SUB-VERBAL)
          Subject :
            {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
                (*WORD* kIrmIzI top) (*HPOSS* NIL)
               {*MODIFIER*
                  {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
                      (*WORD* kIrmIzI) (*HPOSS* NIL)
                     ((*CAT* ADJ)
                      (*R* kIrmIzI))
                  }
               }
               {*MODIFIED*
                  {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
                      (*WORD* top) (*HPOSS* NIL)
                     ((*CAT* N)
                      (*R* top))
                  }
               }
            }
          Dative   :
          DIRECT (*OCC* OPTIONAL) (*ROLE* GOAL)
            {NP (*CASE* DAT) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
                (*WORD* yere) (*HPOSS* NIL)
               ((*CAT* N)
                (*R* yer))
            }
          Predicate :
            {(*TYPE* GERUND) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
                (*WORD* CarptIkCa) (*ROOT* Carp)
               ((*CAT* ADV)
```

```
                (*CONV-FROM*
                        ((*CAT* V)
                         (*R* Carp))
                        (*WITH-SUFFIX* dikce))
                (*SUB* TEMP))
            }
        }
  }
 Direct Object :
 DIRECT (*OCC* OBLIGATORY) (*ROLE* THEME)
   {NP (*CASE* NOM) (*AGR* 3PL) (*POSS* NIL) (*DEF* NIL)
       (*WORD* SarkIlar) (*HPOSS* NIL)
      ((*CAT* N)
       (*R* SarkI))
   }
 Predicate :
   {(*TYPE* VERBAL) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
       (*WORD* sOylUyordu) (*ROOT* sOyle)
      ((*CAT* V)
       (*R* sOyle)
       (*ASPECT* PR-CONT)
       (*TENSE* PAST)
       (*AGR* 3SG))
   }
}

PARSE 3 :

{ SENTENCE (*SUBJ-TYPE* TRANS-ACT-VERBAL)
 Subject :
   {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
       (*WORD* kIrmIzI top) (*HPOSS* NIL)
      {*MODIFIER*
         {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
             (*WORD* kIrmIzI) (*HPOSS* NIL)
            ((*CAT* ADJ)
             (*R* kIrmIzI))
```

```
        }
     }
     {*MODIFIED*
        {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
            (*WORD* top) (*HPOSS* NIL)
           ((*CAT* N)
            (*R* top))
        }
     }
  }
Adverbial Adjunct :
  {ADVP (*TYPE* TEMP)
        { GERUND (*SUBJ-TYPE* ACT-HSUB-VERBAL)
         Dative   :
         DIRECT (*OCC* OPTIONAL) (*ROLE* GOAL)
           {NP (*CASE* DAT) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
               (*WORD* yere) (*HPOSS* NIL)
              ((*CAT* N)
               (*R* yer))
           }
         Predicate :
           {(*TYPE* GERUND) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
               (*WORD* CarptIkCa) (*ROOT* Carp)
              ((*CAT* ADV)
               (*CONV-FROM*
                       ((*CAT* V)
                        (*R* Carp))
                       (*WITH-SUFFIX* dikce))
               (*SUB* TEMP))
           }
        }
  }
Direct Object :
DIRECT (*OCC* OBLIGATORY) (*ROLE* THEME)
  {NP (*CASE* NOM) (*AGR* 3PL) (*POSS* NIL) (*DEF* NIL)
      (*WORD* SarkIlar) (*HPOSS* NIL)
     ((*CAT* N)
```

```
       (*R* SarkI))
   }
 Predicate :
   {(*TYPE* VERBAL) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
       (*WORD* sOylUyordu) (*ROOT* sOyle)
      ((*CAT* V)
       (*R* sOyle)
       (*ASPECT* PR-CONT)
       (*TENSE* PAST)
       (*AGR* 3SG))
   }
}

PARSE 4 :

{ SENTENCE (*SUBJ-TYPE* TRANS-ACT-VERBAL)
 Subject :
   {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
       (*WORD* kIrmIzI top) (*HPOSS* NIL)
      {*MODIFIER*
         {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
             (*WORD* kIrmIzI) (*HPOSS* NIL)
            ((*CAT* ADJ)
             (*R* kIrmIzI))
         }
      }
      {*MODIFIED*
         {NP (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
             (*WORD* top) (*HPOSS* NIL)
            ((*CAT* N)
             (*R* top))
         }
      }
   }
 Adverbial Adjunct :
   {ADVP (*TYPE* TEMP)
         { GERUND (*SUBJ-TYPE* ACT-HSUB-VERBAL)
```

```
          Predicate :
            {(*TYPE* GERUND) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
                (*WORD* CarptIkCa) (*ROOT* Carp)
               ((*CAT* ADV)
                (*CONV-FROM*
                        ((*CAT* V)
                         (*R* Carp))
                        (*WITH-SUFFIX* dikce))
                (*SUB* TEMP))
            }
         }
   }
 Dative   :
 INDIRECT (*OCC* OPTIONAL) (*ROLE* GOAL)
   {NP (*CASE* DAT) (*AGR* 3SG) (*POSS* NIL) (*DEF* NIL)
       (*WORD* yere) (*HPOSS* NIL)
      ((*CAT* N)
       (*R* yer))
   }
 Direct Object :
 DIRECT (*OCC* OBLIGATORY) (*ROLE* THEME)
   {NP (*CASE* NOM) (*AGR* 3PL) (*POSS* NIL) (*DEF* NIL)
       (*WORD* SarkIlar) (*HPOSS* NIL)
      ((*CAT* N)
       (*R* SarkI))
   }
 Predicate :
   {(*TYPE* VERBAL) (*CASE* NOM) (*AGR* 3SG) (*POSS* NIL)
       (*WORD* sOylUyordu) (*ROOT* sOyle)
      ((*CAT* V)
       (*R* sOyle)
       (*ASPECT* PR-CONT)
       (*TENSE* PAST)
       (*AGR* 3SG))
   }
}
```