**T.C.**
**IŞIK UNIVERSITY**
**SCHOOL OF GRADUATE STUDIES**

**DOCTORAL THESIS**
**DEPARTMENT OF COMPUTER ENGINEERING**
**PROGRAM**

**Berke ÖZENÇ**

# AN APPROACH TO ANALYSE TURKISH SYNTAX AT THE MORPHOSYNTACTIC LEVEL

**SUPERVISOR**
**Prof. Ercan SOLAK**

**İSTANBUL, January 2025**

**T.C.**
**IŞIK UNIVERSITY**
**SCHOOL OF GRADUATE STUDIES**

**DOCTORAL THESIS**
**DEPARTMENT OF COMPUTER ENGINEERING**
**PROGRAM**

**Berke ÖZENÇ**

**(217DCS8265)**

# AN APPROACH TO ANALYSE TURKISH SYNTAX AT THE MORPHOSYNTACTIC LEVEL

**SUPERVISOR**

**Prof. Ercan SOLAK**

**İSTANBUL, January 2025**

**T.C.**
**IŞIK UNIVERSITY**
**SCHOOL OF GRADUATE STUDIES**

**DOCTORAL THESIS**
**DEPARTMENT OF COMPUTER ENGINEERING**
**PROGRAM**

**Berke ÖZENÇ**

**(217DCS8265)**

# AN APPROACH TO ANALYSE TURKISH SYNTAX AT THE MORPHOSYNTACTIC LEVEL

Date: 20.01.2025

Thesis Supervisor: Prof. Ercan SOLAK/ Beykoz University

Jury Members: Assoc. Prof. Arzucan ÖZGÜR/ Boğaziçi University

Asst. Prof. İlknur KARADENİZ/ Özyeğin Üniversity

Asst. Prof. Emine EKİN/ Işık University

Asst. Prof. Tuğba ERKOÇ/ Işık University

**İSTANBUL, January 2025**

# ÖZET

## TÜRKÇE SÖZDİZİMİNİN BİÇİM-SÖZDİZİMSEL SEVİYEDE ANALİZ EDİLMESİ YÖNTEMİ

Sözdizim analizi Doğal Dil İşleme alanında kullanılan temel yöntemlerden biridir. Sözdizimsel Analiz işlemi tümce yapısının çeşitli şekillerde analiz edilmesine imkan tanır. Bileşen analizi de bu yollardan bir tanesidir. Bu yöntem tümce yapısını sözcükler ve tamlamalar arasındaki hiyerarşik ilişkiler olarak tanımlar ve ilişkilerin oluşturduğu bu yapıyı ağaç formatında gösterir. Bileşen çözümlemesi, bileşenlerin nasıl birleştiğini ve hangi bileşenleri oluşturduğunu açıklayan bileşen gramerini kullanır. Bu gramerde sözcüklerden tümcenin kendisine kadar olan tüm yapılar bileşenlerce temsil edilir. Tasarımı gereği, bu yaklaşım dilin evrensel özelliklerine odaklanıyor olsa da yaklaşımın odağı hep İngilizce üzerine olmuştur. Bu durum bileşen yönteminin biçimbirim yönünden zengin dillerde biçimbirimlerin sözdizime kattığı detayları kaçırmasına neden olur.

Bu çalışmada, biçimbirim yönünden zengin dillerin için bileşen yöntemindeki zorlukların üstesinden gelen bir uzantı öneriyoruz. Fikirlerimiz Türkçeye için kurgulanmış olsa da Türkçeye benzer herhangi bir dil için de kullanılabilir. Uzantımız bileşen çözümlemesinin biçimbirim düzeyinden başlamasını sağlar. Böylece, biçimbirimsel yapıların sözdizim analizine katılmasını sağlıyoruz ve sözdizime olan etkilerini gösteriyoruz. Önerilerimizi CYK algoritması üzerinde uyguladık. Sözdizim analizinde, özel kurallarla, biçimbirimlerden kaynaklanan muğlaklıkları analiz sürecine dahil ediyoruz. Ek olarak, Türkçe için, biçimbirim odaklı bir bileşen kümesi tasarladık. Tasarladığımız bu küme ekleri, gövdeleri ve başı gövde olan tamlamaları içerir. Çalışmamızı küçük bir ağaç bankası ve ondan üretilen gramer ile gösteriyoruz.

**Anahtar Kelimeler:** Türkçe, Bileşen Analizi, Bileşen Grameri

# ABSTRACT

## AN APPROACH TO ANAYLSE TURKISH SYNTAX AT MORPHOSYNTACTIC LEVEL

Syntactic analysis allows us to analyse the sentence structure in various ways. Constituency parsing is one of the various ways of conducting syntactic analysis. This parsing method defines sentence structure as hierarchical relationships between words or phrases and represents them in tree form. Constituency parsing employs constituency grammar which defines how constituents combine and form other constituents. In this grammar, any syntactic structure from the sentence to the words is represented by the constituents. Although this approach is designed to focus on universal aspects of the languages, English has always been in its focus. This situation makes the constituency approach miss the details that the morphology puts in the syntax of morphologically rich languages.

In this study, we implement an extension for the constituency parsing which overcomes the challenges in parsing of MRL (Morphologically Rich Language). We propose ideas tailored to Turkish, yet they can be used for any language like Turkish. Our extension enables the constituency parsing to start at the morpheme level. Thus, we involve morphemic structures in the parsing process and express their syntactic effects on the structure. We have our implementations by extending the CYK (Cocke Younger Kasami) algorithm. During parsing, we utilize extra rules to transfer the ambiguity in morphology to the parsing. In addition, we designed a morpheme-focused constituency set for Turkish. This set involves affixes, stems and phrases headed by a stem. We demonstrate our work with a mini treebank and the grammar generated from it.

**Keywords:** Turkish, Constituency Parsing, Constituency Grammar

# ACKNOWLEDGEMENT

First and foremost, I would like to express my thanks and gratitude to my supervisor and teacher Prof. Ercan Solak, who has been supporting me with all his support, guided me with his knowledge and experience and helped me reach this stage. His support, patience and guidance have always been very essential for me.

Also, I would like to express my deepest gratitude to the thesis progression committee members Assoc. Prof. Arzucan Özgür and Asst. Prof. İlknur Karadeniz for their valuable opinions and contributions during this study.

Finally, I would like to thank my wife Sevde Ceren Yıldız Özenç, who has been my greatest support, and my family. I am grateful for their constant support and the belief that they have in me. It was their endless support that kept me going all the time. I am grateful to them for not leaving me alone in this process and keeping me going even in the most difficult moments.

Berke ÖZENÇ

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABBREVIATIONS LIST

**CL:** Computational Linguistics

**NLP:** Natural Language Processing

**CFG:** Context Free Grammar

**CYK:** Cocke Younger Kasami

**SVO:** Subject Verb Object

**SOV:** Subject Object Verb

**RHS:** Right-hand side

**LHS:** Left-hand side

**CNF**: Chomsky Normal Form

**MRL**: Morphologically Rich Language

# CHAPTER 1

## 1. INTRODUCTION

### 1.1 GRAMMAR AND PARSING IN NATURAL LANGUAGE PROCESSING

Computational Linguistics (CL) and Natural Language Processing (NLP) are two related study fields. In recent decades both have gained a significant focus due to the increase in the need for machines to understand, process and generate language. Processing vast amounts of textual language-related data becomes crucial. Such an ability is compulsory for various applications such as machine translation, information extraction, sentiment analysis, etc. The core of these applications is analysing the grammatical structure of the sentence, which is called syntactic parsing (Klein & Manning, 2003).

From a broad perspective, parsing can be explained as breaking down a sentence into pieces to understand and express its syntactic structure. Additionally, it enables more advanced language processing tasks. Generally, parsing can be classified as dependency and constituency parsing. While dependency parsing examines a sentence in terms of the dependency relations between the words, constituency parsing examines the hierarchy of the words.

Along with the parsing, a language model is essential. This is called a syntax model as well since the purpose is to define the rules of the language. Several popular approaches for modelling are constituency grammar, dependency grammar, lexical functional grammar, etc. (Jurafsky & Martin, 2024). The constituency grammar is the specific model to conduct constituency parsing.

## 1.2 CONSTITUENCY GRAMMAR AND PARSING

Constituency grammar is based on the phrase structure grammar. This grammar notion represents the sentences as nested structures (Chomsky, 1965). These structures are called constituents, and they represent specific syntactic categories, phrases and words. Also, there are production rules which define the relationship of the constituents. Particularly, how they are combined to produce which constituent. This enables the sentence to be represented in a tree form, which defines the syntactic structure through the hierarchical relationships between the words and phrases. The most common application of constituency grammar is context-free grammar.

Constituency parsing, on the other hand, is a process to yield this structure through grammar. Additionally, it answers the query of whether a sentence belongs to a language. It has practical applications in various NLP tasks such as machine translation (Yamada & Knight, 2001) and information extraction (Collins, 1997).

There have been a diverse implementations of constituency parsing. Initially, there were rule-based implementations (Earley, 1970). Along with the probability and statistics involved, new grammars were introduced such as Probabilistic Context-Free Grammar that yields statistics-based constituency parsers (Charniak, 2000; Collins, 1997). Along with the improvements in machine learning, learning-based parsers have appeared (Socher et al., 2013).

Considering the environment and the native languages of many researchers, it is natural that the constituency technique is developed under the roof of English. Also, rules, constituent categories and data sets which are used in constituency parsing are generally developed for languages like English. Unfortunately, such a structure cannot be easily ported to a MRL.

## 1.3 SCOPE OF THE THESIS

Turkish is a morphologically rich and agglutinative language with a scrambling word order. The extensive usage of affixes in Turkish enables formation of new words and encoding semantic and syntactic information by just attaching several affixes (Göksel & Kerslake, 2004). The complexity of Turkish morphosyntax challenges syntactic parsing. Especially, when the design of significant grammar and parsing paradigms such as constituency focuses only on a lightly inflected languages such as English. This situation makes the current implementations inadequate for use in Turkish and difficult to port.

The agglutinative word forms and the relatively free word order of Turkish are two crucial challenges in terms of the constituency. Any morphologically rich language possesses similar challenges for constituency parsing. Any implementation of constituency parsing which focuses on such a language needs to overcome those challenges. The grammar adapted for any language like Turkish must involve the morphological structures (Koç, 1990; Underhil, 1985).

In this study, we focus on the constituency parsing of Turkish. We implement an extension the CYK (Cocke Younger Kasami) algorithm to include morphological elements in the parsing process. To enable such an extension, instead of a list of words, we input a list of morphemes to the CYK algorithm. We employ temporary rules to include the morphological ambiguity in parsing and yield all valid parses in terms of both syntactic and morphological aspects. We use filtering to minimize the confusion caused by the homonyms between words and affixes. Additionally, we design a set of constituent and grammar rules which specifically tailored for Turkish. Our set of constituents is designed to meet the requirements of the effects of morphology in the syntax. In this set, we considered stems, affixes and stemmed phrases along with the words and we aimed to obtain the minimal set of constituents and rules.

# CHAPTER 2

## 2. LITERATURE

### 2.1 RELATED STUDIES

#### 2.1.1. Studies Related To Other MRL

In the literature, there are several studies focusing on constituency parsing of other MRLs. All these studies involve morphology in parsing.

An Early based constituency parser is developed in (Abbas, 2016). This parser utilizes a set of constituents which consists of three parts: semi-semantic part of speech (POS) tags, semi-semantic syntactic tags and functional tags. To include morphological features in the parsing process, the traditional POS tags are expanded based on the morphological aspect of each token. Additionally, in this study, a modified Early algorithm is proposed where the modifications enable the elimination of unnecessary comparisons done on terminal symbols. Also, back pointers are added into the process thus they enable the algorithm to yield trees of each parse.

Another study focuses on developing both constituency and dependency parsing for Modern Hebrew (Goldberg, 2011). The constituency parsing in this study employs the traditional CYK algorithm. Also, the traditional set of constituents are categorized based on morphological and syntactic features.

Babarczy et al. (2005) propose a rule-based constituency parser for Hungarian. The parser is designed to recognize the syntactic elements based on their morphological cues. Also, the rules of the grammar used in the parser involve morphological features. Each input token is put through the morphological analysis. The analysis result is then used to match the morphological aspects of the tokens and the rules. Thus, the system can build up the structure of the sentence. To improve the identification of tokens, the

parser also utilizes a lexicalized database. The steps of the parsing process include identifying the certain phrase type and its complements by rule matching.

For German, a statistical CYK parser is developed in (Fraser et al., 2013). The parser employs bit vectors and uses a grammar extracted from Tiger2 Treebank (Brants et al., 2002). The treebank consists of trees with annotations done based on syntactic and morphological categories together. In the study, they also compare the effects of several features like lexical and morphological information on constituency parsing. The authors report an increase in parsing performance due to the usage of morphological cues in parsing.

A recent study aims to reduce the error rate of constituency parsing in Korean (Kim & Park, 2022). In the study, the authors measure the performance of different parsing algorithms on different grammars. The grammar which they use in comparison is generated from a morpheme-based treebank. They report increased performance and decreased error rate for the morpheme-based grammar.

### 2.1.2. Studies Related To Turkish

In the NLP, Dependency grammar (De Marneffe & Nivre, 2019; Tesnière, 2015) has been the popular study topic for syntactic analysis of Turkish. Dependency grammar defines a parse as a series of binary dependency relations between the words. In each dependency relationship, there is a head and its complement. The head is the main element in the relation while the complement qualifies the head in terms of different syntactic roles. There are several studies which take the dependency parsing of Turkish into focus (Eryiğit et al., 2008; Tuç, 2020; Türk et al., 2022). The popularity of dependency grammar is similar in treebank-focused studies (Çöltekin, 2015; Kayadelen et al., 2020).

There are fewer studies which address the constituency grammar and constituency treebank construction from a morphological perspective.

A recent study proposes a context-free grammar for Turkish (Dönmez & Adalı, 2018). In this grammar, the phrases are categorized based on their

syntactic role, structure and affixes that they take. Also, the rules in the grammar are diversified based on the affixes attached to the structure.

Another paper which studies constituency grammar for Turkish, addresses the topic from the treebank perspective in (Yildiz et al., 2015). Authors construct a treebank consisting of parse trees which involve morphological annotations.

In the NLP area, besides the constituency, there are studies that focus on different grammar methodologies and include morphology in syntax. A combinatorial categorical grammar is proposed in (Bozsahin, 2002), where the grammar includes morphemes and defines the combinatorial behaviour of morphemes and words. In another study, a lexical function grammar is proposed where the grammar is designed to focus on the morphological cues (Çetinoğlu & Oflazer, 2018).

There are a number of machine learning-based parsing studies which combine syntactic approaches and morphology, besides the rule-based applications. The dependency parsing approach is still popular, so that, in learning-based studies, it is a frequently used idea to involve morphology in dependency parsing (Çakıcı, 2009; Can et al., 2022; Çetinoğlu & Kuhn, 2013; Eryigit, 2012). Additionally, to increase the accuracy of parsing highly inflected languages, a parsing-tagging hybrid model involving dependency parsing and part-of-speech tagging employs morphological analyses (Bohnet et al., 2013). A transition-based parsing system is extended with morphological structures (More et al., 2019). The system unifies the morphological and dependency rules in the transition system.

Even though dependency parsing is a widely utilized method in parsing studies for Turkish, through an extra conversion step, dependency-based techniques may be duplicated using constituency-based techniques. Despite the differences between constituency and dependency techniques, constituency parses may be deterministically converted to dependency ones (Johansson & Nugues, 2007).

In purely linguistics studies in the literature, we can see that utilising morphemes as a part of the syntax has already been an idea adopted for Turkish grammar.

Koç (1990) states that the atomic elements which form a sentence may be morphological elements. In his book, the functions of morphological elements in the Turkish syntax are explained through examples. The illustrations of sentence structure show that the attachment of verbal affixes such as tense or person joins the structure individually, not in a way that is attached to the verb stem or root. Similarly, illustrations showing noun phrases show that case affixes join the structure separately from the noun stem or root that they are attached to. Other sources share similar ideas (Uzun, 2000).

In another study, The sentence structure of Turkish is analysed as in English (Underhil, 1985). The author points out that there is an auxiliary verb in Turkish sentences as in English. However, unlike in English, Turkish auxiliary is a morpheme or a morpheme group. Additionally, the author illustrates that several morphemes exist in the sentence structure separate from the word roots.

Although the fundamental idea we have in this study is like what those previous studies illustrate, our work shows differences in terms of the definition of the verbal phrase, the structural difference in the combination of the verb and its complements, groups of morphemes and the functions of case marked nouns.

Similar ideas can be observed in various linguistics studies as in structural variations in verbal inflection, structure of case-marked complex noun phrases or affixed subordinate clauses (Erkman-Akerson, 1994; Göksel & Kerslake, 2004; Hankamer, 2004; Karabulut, 2009; Kornflit, 1996; Malkoç, 2011).

# CHAPTER 3

## 3. TRADITIONAL GRAMMAR AND PARSING

### 3.1 CONSTITUENCY GRAMMAR

Constituency grammar is based on the idea that sentences are composed of units called constituents. The constituents can be groups of words which function together as a unit or a single word. This allows constituents to be nested within a larger constituent. In this way, it defines the hierarchical a sentence, (Jurafsky & Martin, 2024).

The fundamentals of constituency grammar are based on Noam Chomsky's notion of the phrase structured grammar where words are combined to form phrases and sentences. Similarly, in constituency parsing, constituents represent words, phrases and sentence on certain levels of the sentence structure.

Constituents are classified into several categories in this grammar formalism and these categories are mostly common to all languages. Some examples are Nominal (N), a category for nominal words such as 'dog' or 'man', Determiner (Det), a category for determiner words such as 'the', Nominal Phrase (NP), a category for noun phrases such as 'the man' or 'the dog', Verb (V) such as 'bit' and verb phrases such as 'bit the man' and Sentence (S) such as 'the dog bit the man'. Examples already show the nested structure of the constituents but there is a better way to see the overall structure.

The hierarchical structure of the sentence is represented as a tree diagram which illustrates how constituents are nested. In Figure 3.1, the tree diagram illustrates the structure of the sentence 'the dog bit the man'.

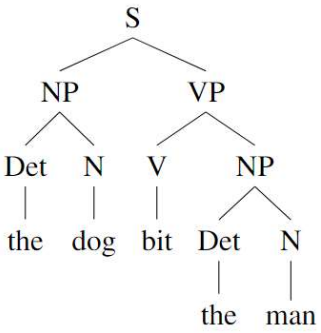


**Figure 3.1** The tree diagram for the sentence 'the dog bit the man'.

## 3.2 CONTEXT-FREE GRAMMAR

A common implementation of constituency grammar is Context-Free Grammar (CFG). This formalism inherits the aspects of the constituency grammar and implements them regardless of the context.

A CFG consists of symbols and production rules which define how symbols can be combined to generate a valid sentence. A production rule is in the following form.

(3.1) $A \rightarrow B_1 B_2 B_3 \ldots B_n$

Where A is a non-terminal symbol and Bs are either non-terminals or terminals. In this formalism, non-terminals are the constituents and terminals are the actual words of the sentence. Production rules are also referred to as grammar rules or rewrite rules. In the rest of the report, we will refer to them as grammar rules.

Since both A and Bs can be constituents, some grammar rules define constituents as a combination of other constituents. This reveals the recursive nature of CFGs. With the recursive structure, a CFG can generate an infinite number of sentences from a finite set of grammar rules, (Chomsky, 1956).

To emphasize the connection between CFG and constituency, we can build the grammar that generates the sentence 'the dog bit the man'. In fact,

induced rules of such a grammar are illustrated in Figure 3.1 in the tree form. Such grammar is given below.

(3.2) $S \rightarrow NP\ VP$

$NP \rightarrow Det\ N$

$VP \rightarrow V\ NP$

$V \rightarrow bit$

$Det \rightarrow the$

$N \rightarrow dog \mid man$

## 3.3 CONSTITUENCY PARSING

The process of determining whether a sentence can be generated from a certain grammar is called parsing. If the grammar in the question is a Constituency Grammar, the process is specifically called Constituency Parsing. In the rest of the thesis, the word parsing is used to mean constituency parsing.

Parsing is referred to as syntactic analysis as well because, in addition to answering the membership issue, it also aims to yield certain structure or properties of the sentence with respect to the type of grammar used. Particularly for constituency parsing, the aim is to determine the constituent structure of the sentence and to construct parse trees to represent the hierarchical structure of the sentence.

There are two main approaches to constituency parsing. top-down and bottom-up parsing.

The top-down approach starts the process at the highest level of the sentence hierarchy. Working by the grammar rules, it identifies constituents by decomposition as it works down on the structure until it finds the words of the sentence as terminals.

The bottom-up approach, on the contrary, starts at the lowest level of the hierarchy. Starting from the words, it goes up, as it combines constituents by the grammar rules until the constituent S is formed on the top level. In this dissertation, we study CYK algorithm.

## 3.4 CYK ALGORITHM

The CYK algorithm is a parsing technique used with context-free grammar (Cocke & Schwartz, 1970; Kasami, 1966; Younger, 1967). The algorithm determines if the given grammar can generate the given string. If it can, then the CYK produces all possible parse trees of the input string. In general, the input string corresponds to a sentence.

The algorithm decomposes the overall parsing process into smaller steps. Each step involves merging two constituents into a bigger one by following grammatical rules. The CYK starts to process from the smallest constituents. Therefore, the initial steps combine the tokens, if we assume the input is a sentence. It continues to take steps in a bottom-up manner until the given string is acquired on a certain grammatical level.

This approach requires the given context-free grammar to be in Chomsky Normal Form (CNF), (Chomsky & Miller, 1963). CNF is a specific form of grammar where each rule is in the form of $A \rightarrow B\ C$ or $A \rightarrow w$ where $A$, $B$ and $C$ are non-terminals and $w$ is a terminal. Any grammar must be put in CNF before it is used in the CYK algorithm.

The CYK algorithm is a dynamic programming approach that involves a two-dimensional matrix, (Jurafsky & Martin, 2024). It is also called a chart parsing algorithm and there are various implementations. Depending on the implementation, a matrix of size $N \times N$ or $(N+1) \times (N+1)$ is constructed for a sentence of $N$ tokens. The algorithm uses the upper or lower triangular half of the matrix. Each cell may contain a set of constituents. Considering the variation that uses the lower-triangular half of the $N \times N$ matrix, a constituent at the cell $[i-1, j-1]$, must be combined by the constituents at cells

$[k-1, j-1]$ and $[i-k-1, j+k-1]$ respectively where $1 \leq k < i$. Any combination is done by following a corresponding grammar rule. The combined constituents must match with the right-hand side of the grammar rule. The left-hand side constituent of the rule is placed in cell $[i, j]$ as a result of the combination. The table is filled from bottom to up. The terminal symbols are not included in the table but using terminal rules of the grammar, their part of speech (POS) tags are placed into bottom cells. Every other cell is filled by following the non-terminal rules of grammar.

The table of the CYK algorithm during parsing of the sentence 'The dog bit the man' is illustrated in Figure 3.2. Parsing this sentence requires the minimal grammar given in (3.2).

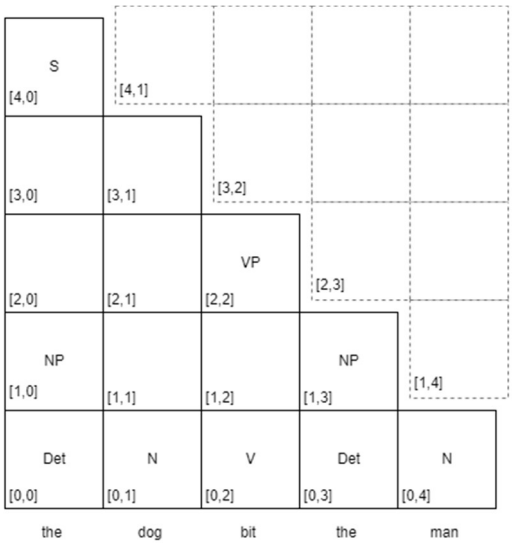


**Figure 3.2** Parsing table for the sentence "the dog bit the man"

Filling the table is enough to recognize a sentence. The algorithm returns all possible parse trees. Returning all found roots is enough but the root and the other nodes of the tree need to point each other hierarchically. Adding back pointers for each constituent in each cell will be enough to achieve such a structure. For every constituent, the back pointer indicates a specific constituent at a specific index.

## 3.5 TRADITIONAL CONSTITUENCY APPROACH AND TURKISH

The constituency approach was developed under the influence of English. Even though the fundamental of the approach conforms to the universal structures of a language, it primarily focuses on the problems inherent to English.

The rule structure of constituency grammar explains it well. Each rule in constituency defines which constituents are combined into which one. In addition to that, it specifies the order of the constituents as well. This means that the grammar rules can only validate the matching order of the words of the sentence.

The word order has a significant role in English syntax, and it determines the function of the syntactic element. The dominant order is Subject-Verb-Object (SVO). It specifies that the verb must be in the middle of the sentence, and the subject and the object must be positioned on the left and the right respectively. Consider the sentence "the dog bit the man" again. In this sentence, "the dog" is the subject (who does the action), "bit" is the verb (the action itself) and "the man" is the object (the one affected by the action). These three roles are assigned to the elements by their order and the order is recognized by the grammar rules as we discussed previously. When you change the order as in "the man bit the dog", although now there is a new odd meaning and a new sentence, it is recognized as well. This is because the elements which interchange their positions are both nominal phrases and they both can fit into subject and object functions. So, the new sentence does not violate any grammar rules. Out of any context, we can summarize the constituent order for English as NP V NP and this order can be derived from the grammar given in (3.2). Any order which does not conform to this will not be recognized by the grammar rules. For example, "the dog the man bit". In terms of functions, the word order can be expressed as Subject-Object-Verb (SOV) and in terms of constituents, it is NP NP V.

The fact that the grammar rules determine the order is a challenge for any language which has relatively free word order such as Turkish. Unlike English, the word order does not determine the function of the elements in Turkish. It determines the information structure of the sentence in terms of topic, focus and background, (Erguvanli, 1984). Changing the word order yields distinct and semantically different meanings of the same sentence. Nevertheless, it does not yield an invalid sentence.

The sentence 'the dog bit the man' can be translated into Turkish as in (3.3).

For both sentences, the elements and their functions are equivalent, yet the order is different. The unmarked (dominant) word order for Turkish is SOV. Although this is an invalid word order in English, it is valid and the most common one in Turkish. In addition, by changing the word order only, the same sentence can be formed in six different ways. All of them are valid sentences for Turkish.

(3.3) köpek adam-ı ısır-dı
dog man-<Case:Acc> bite-<Tns:Past><Prsn:3s>
'the dog bit the man'

Another challenge is the depth that the constituency approaches reach. English has a simpler morphology compared to Turkish. Therefore, it is natural that the constituency approach which focuses on English, does not involve any sub-word structures in the syntactic processes.

In Turkish, syntactic functions are assigned to elements by attaching specific morphological cues, contrary to the one based on the word order in English. For example, regardless of the position, a nominal takes object functionality if it is attached to the Accusative case morpheme such as 'adam-ı' in (3.3). Additionally, the rich morphology of Turkish allows the production of new words from the roots which have varying types. Since the morphemes have such effect, a grammar which involves roots and morphemes is essential in order to have a comprehensive syntactic analysis of such a language.

# CHAPTER 4

## 4. MORPHOSYNTACTIC CONSTITUENCY GRAMMAR AND TREEBANK

### 4.1 MORPHOSYNTACTIC CONSTITUENCY GRAMMAR

Grammar is a language model. A parser needs grammar to parse sentences. This study focuses on constituency grammar. Constituency grammar consists of two elements, constituents and rules. Constituents represent syntactic elements such as a sentence, a phrase or a word. Rules express how these constituents combine and what they form in terms of the constituency. In this study, in addition to the morphosyntactic parsing algorithm, we study a grammar based on constituency grammar formalism and integrate morphology and syntax. This section gives the details of our grammar.

Like traditional constituency grammar, our grammar expresses sentences hierarchically. This hierarchical structure is called a parse tree. The sentence appears at the highest level of the hierarchy and is represented via the constituent S. It is called the root of the tree. Via the grammar rules, the root is broken into smaller constituents, on one level below. The smaller constituents are also broken into much smaller constituents in the same manner. Constituents of the intermediate hierarchy levels are called non-terminals. This continues recursively until the bottom level of this hierarchy. The bottom-level constituents are called terminals.

In our grammar, a terminal corresponds to a word stem, a suffix or a group of suffixes. For the non-terminals, we developed a novel tag set based on the following assumptions.

1. A valid parse tree must have a single root, S.
2. Standalone morphemes, groups of morphemes, word roots or stems may form a leaf of a syntax tree.

3. If two constituents have different combinations based on the context they have distinct tags.
4. Phrases that qualify a noun or a verb attach to the stem instead of the inflected noun or the finite verb. Any further inflectional suffixes are attached after the qualifying attachments.
5. Punctuations are not included in the grammar.
6. Morphological identifiers are used as Part of Speech tags.

Based on these rules, we designed a constituent set to be used in morphosyntactic parsing. Additionally, we tailored the set with respect to the syntactic and morphological features of Turkish. The constituent set that we designed consists of two types of constituents, morpheme and phrase constituents. Morpheme constituents are used for word stems, morphemes or groups of morphemes, while phrase constituents are used for finite and stemmed phrases.

In the following parts of this section, firstly, we will explain the two unique aspects, stemmed phrase and morphological identifiers, in Sections 3 and 4.1.2 respectively. Next, we will give the details for each constituent, and with examples, we will explain the structures that they exist in.

### 4.1.1. Stemmed Phrase

The stemmed phrase is an intermediate constituent which appears during the formation of a phrase. It represents an incomplete phrase which contains a word stem, qualifier phrase and morphemes, at the simplest construction. In the rule 4, we stated that the qualifiers are attached to the word stems, directly. We call the structures that are formed due to the fourth rule, stemmed phrases. We use the XPS (X stemmed phrase) constituent to show the stemmed phrase which belongs to an X type of phrase.

A stemmed phrase can be formed in two different ways.

A stem which takes several morphemes that do not belong to the same constituents may form a stemmed phrase. In this case, morphemes are attached to the stem due to the morphological order. The first morpheme in the order is

attached to the stem directly and they form a stemmed phrase. The following morphemes are attached to stemmed phrases. Each morpheme which belongs to a different constituent, forms a new stemmed phrase, except the last one in the order. The last morpheme completes the stemmed phrase into a phrase. An example phrase is given (4.1) to illustrate this structure. The simplified syntax tree which belongs to the phrase in (4.1) is illustrated in Figure 4.1. In the example, the word 'taksi', takes the plural suffix first and they form NPS (nominal phrase stem). Next, the locative case suffix is attached to the structure and another NPS is formed. When the last suffix of the word, the relativizer suffix, is attached to the structure, the nominal phrase NP (nominal phrase) is formed.

(4.1) taksi-ler-de-ki

taxiz<NOM>-<Num:Pl>-<Case:Loc>-<Rel><NOM>

'one that is in taxis'


**Figure 4.1** The parse sub-tree of the phrase in (4.1).

Additionally, a stemmed phrase is formed when a qualified word is inflected. In this structure, the qualifier and the word stem form a stemmed phrase. Next, morphemes of the stem are attached to the stemmed phrase in the same way shown above. A nominal phrase example consists of an inflected word and a qualifier is given in (4.2). The parse sub-tree belonging to the phrase is shown in Figure 4.2.

(4.2) sarı taksi-ler-de-ki

yellow<NOM> taxi<NOM>-<Num:Pl>-<Case:Loc>-<Rel>

'one that is in yellow taxis'

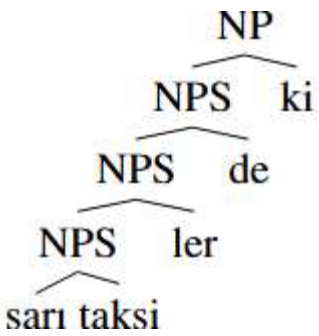


**Figure 4.2** The parse sub-tree of the phrase in (4.2).

We simplified the constituents and the parse trees in the above examples to focus on the stemmed phrase structure. We will explain the details of nominal constituents in Section 4.1.4.

### 4.1.2. Morphological Identifiers

Both the grammar and the analysis method we propose are based on morphemes. Each morpheme has an abstract and a surface form. While surface form defines how the morpheme is read and written, abstract form is an identifier which indices where this morpheme occurs in the morphology, what comes before or after it, etc.

It can be difficult to identify a morpheme because of the homonyms and alternations that they have through phonologic rules. This becomes more of an issue when the morpheme has zero surface form. Identifying the nature of abstract forms aids us even though the morpheme has no surface form.

A word may have several analyses. All such analyses must be processed in order to obtain all possible parses on the morphosyntactic level. Unfortunately, different morphological analyses vary in terms of the number and position of morphemes. This situation brings a challenge to our parsing

method. We can illustrate this problem with an example. Table 4.1 shows the analysis of ‘dişi’ where each morpheme is in an individual cell and empty cells indicate zero surface forms.

**Table 4.1** Analyses of ‘dişi’

| | | | | | |
|---|---|---|---|---|---|
| **Analysis 1** | **Surface** | diş | | i | |
| | **Abstract** | <NOM> | <Num:Sg> | <NC> | <Case:Nom> |
| **Analysis 2** | **Surface** | diş | | i | |
| | **Abstract** | <NOM> | <Num:Sg> | <Poss:3s> | <Case:Nom> |
| **Analysis 3** | **Surface** | diş | | | i |
| | **Abstract** | <NOM> | <Num:Sg> | <Poss:No> | <Case:Acc> |
| **Analysis 4** | **Surface** | dişi | | | |
| | **Abstract** | <NOM> | <Num:Sg> | <Poss:No> | <Case:Nom> |

Although the morpheme positions vary, this will not be a problem if we just focus on the ones with the surface forms only. From the surface perspective, the situation looks more stable. Table 4.2 shows the surface morphemes in analyses of ‘dişi’. In this case, morphemes match in both the number and position except the last one. The exception of the last morpheme is not related to zero morphemes therefore we will neglect it for now. This example shows that we must handle the zero-form morphemes to have a more stable morpheme list.

**Table 4.2** Surface morphemes in Analyses of ‘dişi’

| | | | |
|---|---|---|---|
| **Analysis 1** | **Surface** | diş | i |
| | **Abstract** | <NOM> | <NC> |
| **Analysis 2** | **Surface** | diş | i |
| | **Abstract** | <NOM> | <Poss:3s> |
| **Analysis 3** | **Surface** | diş | i |
| | **Abstract** | <NOM> | <Case:Acc> |
| **Analysis 4** | **Surface** | dişi | |
| | **Abstract** | <NOM> | |

Even the zero morphemes indicate specific information about the word. So, they cannot just be deleted. Instead, we group them with the ones which have non-zero surface form. Considering the morphotactic rules, a mindless grouping can only harm the structures and the parsing system. Therefore, we set the following rules to group zero morphemes with non-zero ones. Naturally, the grouping is done based on the surface forms.

- If there are both zero and non-zero morphemes which belong to the same group, abstract forms of all zero morphemes are attached to the abstract form of the non-zero one.
- If the morphemes belong to the same group and are all zero morphemes, all abstract morphemes of zero morphemes are attached to the closest non-zero morpheme on the left.

Based on these rules, we can revisit the analyses of 'dişi' and obtain the structures given in Table 4.3.

**Table 4.3** Analyses of 'dişi' where morphemes are grouped

| | | | |
|---|---|---|---|
| **Analysis 1** | **Surface** | diş | i |
| | **Abstract** | <NOM><Num:Sg> | <NC><Case:Nom> |
| **Analysis 2** | **Surface** | diş | i |
| | **Abstract** | <NOM> | < Num:Sg ><br><Poss:3s><Case:Nom> |
| **Analysis 3** | **Surface** | diş | i |
| | **Abstract** | <NOM>< Num:Sg ><Poss:3s> | <Case:Acc> |
| **Analysis 4** | **Surface** | dişi | |
| | **Abstract** | <NOM>< Num:Sg ><Poss:No><Case:Nom> | |

We use morpheme grouping in grammar, parsing and treebank construction stages. In the rest of the examples, zero morphemes are grouped regardless we if show them.

### 4.1.3. Verbal Constituents

The top-level constituent S represents the sentence itself both in the grammar and a parse tree. S appears as the root of the parse trees, with several exceptions. Generally, S is formed by a verb phrase for a simple, verbal sentence. We show verb phrases with the constituent VP (verb phrase) in the grammar.

The verb phrase structure includes the verb stem, verbal suffixes such as negation, tense and person, and verb qualifiers. Since the qualifiers are optional, only the verb stem, tense and person suffixes are enough to form a sentence with minimal construction. We show the verb stems with the constituent VS (verb stem). We grouped the tense, person and copula suffixes, and showed them with TPMG (tense person morpheme group) constituent. A sentence which consists of a minimal number of constituents is given in (4.3). The parse tree of the sentence is shown in Figure 4.3. In the example, the verb stem *gel* 'come' takes the tense and person suffix group. They form the verb phrase. Next, the verb phrase forms the sentence.

(4.3) gel-di-m

come<VS>-<Tns:Past>-<Prsn:1s>

'I came'

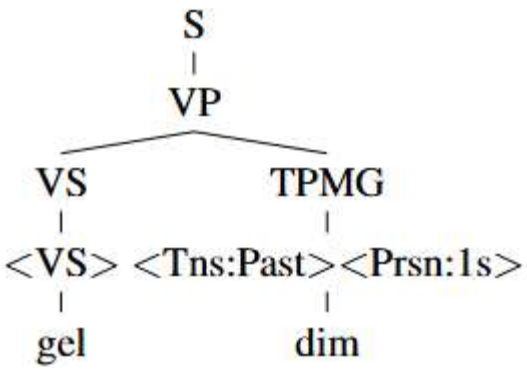


**Figure 4.3** The parse tree of the sentence in (4.3).

If the verb reports negative polarity or it takes any passive, causative, ability or probability suffixes, they are attached to the verb stem as a group of verbal morpheme suffixes (VMG). The negative version of (4.3) is given in (4.4) and the parse tree of the negative sentence is shown in Figure 4.4. In such a construction, VS and VMG are combined into VPS (Verb phrase stem) first. Then VPS is combined with TPMG to complete the verb phrase.

(4.4) gel-me-di-m

come<VS>-<Pol:Neg>-<Tns:Past>-<Prsn:1s>

'I did not come'

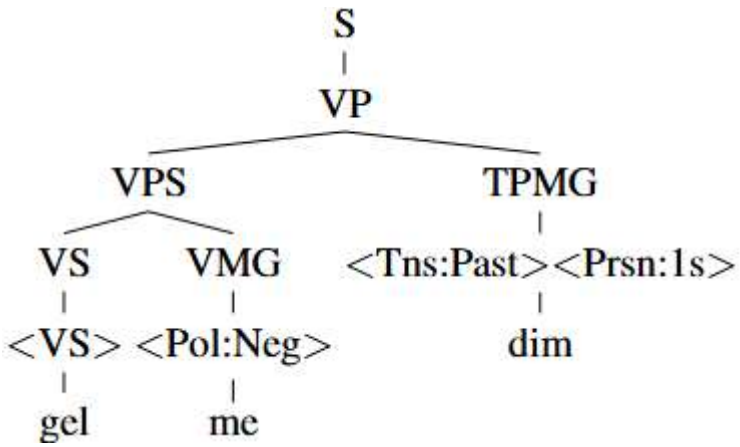


**Figure 4.4** The parse tree of the sentence in (4.4).

Instead of the verb stem, a derived verb can form a verb phrase. Regardless of the initial type of the word, derivation results as the VPS constituent. We show the suffixes which derive verbs with the constituent VDM (verb-deriving morpheme). An example sentence which is formed by a derived verb is given in (4.5) and the parse tree of the sentence is given in Figure 4.5. In the example, the verb is derived from a nominal stem. We represent the nominal stems with the NS (nominal stem) constituent. Details about nominal constituents are given in Section 4.1.4.

(4.5) insan-laş-tı-m

human<NOM>-<Bec><VS>-<Tns:Past>-<Prsn:1s>

'I became human'

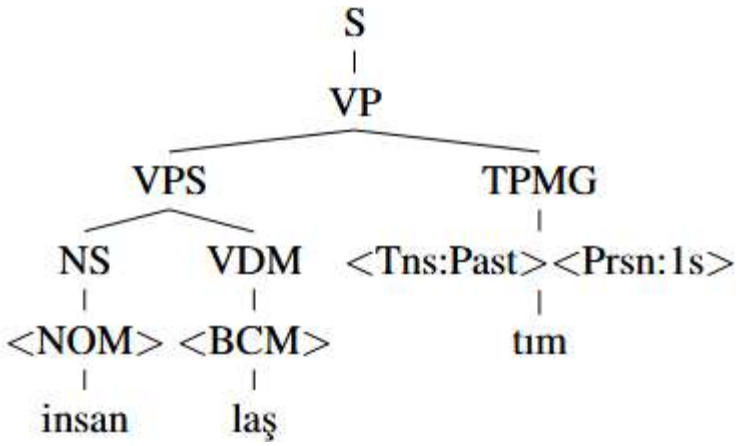


**Figure 4.5** The parse tree of the sentence in (4.5).

An adverb phrase can be included in a more complex verb phrase structure. In the grammar, we show the adverb phrases with the ADVP (adverb phrase) constituents. As with other qualifiers, adverb phrases are combined with the stem of the verb for which they qualify. We show the qualified verb stem with the VPS constituent. The verb stem takes morphemes after the qualification. An example sentence which contains an adverb phrase is given in (4.6). The parse tree of the example sentence is illustrated in Figure 4.6.In the example, the verb stem is qualified by a derived adverb. We omit the adverb phrase structure to keep the parse tree simple. We give the details related to adverb structures in Section 4.1.5.

(4.6) hızlıca<ADV> gel-di-m

fast<ADV> come<VS>-<Tns:Past>-<Prsn:1s>

'I came fast'

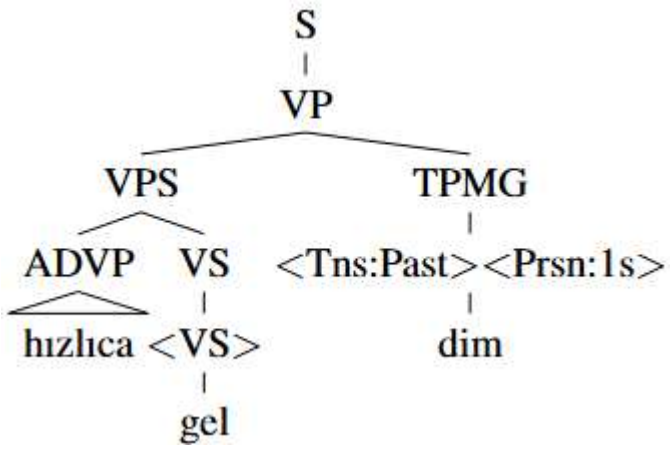


**Figure 4.6** The parse tree of the sentence in (4.6).

Verb phrase structure may contain syntactic elements such as object and subject. Generally, they are formed by nominal constructions. Therefore, we explain the nominal constituents before including these elements in the verb phrase structure.

### 4.1.4. Nominal Constituents

We show a nominal phrase with the constituent NP. Considering the different structures that the nominal phrases are formed by and the different features that they possess in the sentence, NP and the constituents which form NP require several categorizations. In this section, we give the details of nominal phrase structures.

Due to the nominal inflection order, first, the number morpheme, then, the possessive morpheme and lastly the case morpheme is attached to the word. Each one of these tree morphemes may have a non-overt form which requires the categorization of the nominal stem constituent.

A singular nominal which has no possessor and is marked with the nominative case passes through the nominal inflection without taking any overt morphemes. A nominal stem having this specific inflection may form a nominal phrase on its own. We categorize such stems with the constituent NS3. An example phrase is given in (4.7). The parse sub-tree of the phrase is shown in

Figure 4.7. We limited the examples which we give for NS categorization, to the nominal phrase instead of taking a full sentence to emphasize the structure. In this example, we particularly show the abstract forms of the zero morphemes. For the further examples, we will omit them to keep the focus on the structure we explain.

(4.7) kedi

cat<NOM><Num:Sg><Poss:No><Case:Nom>

'cat'

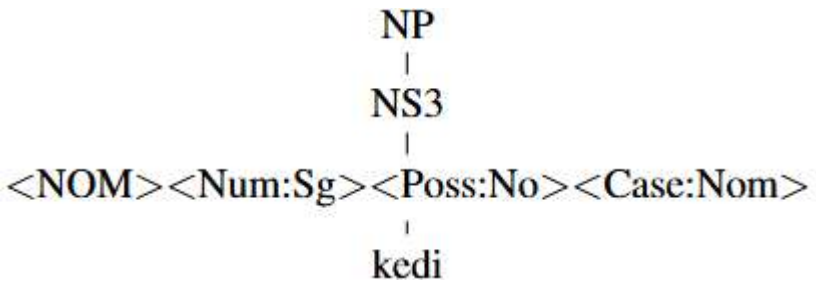


**Figure 4.7** The parse sub-tree of the phrase in (4.7).

We group the overt forms of the number and possessive morphemes because they show the same behaviour when only one of them is overt or both are overt. We show this morpheme group with the PLPMG (plural and possessive morphemes group) constituent. We combine PLPMG with NS1. NS1 represents the nominal stem which takes plural and/or possessive morphemes. NS1 and PLPMG combination forms NPS1 constituent which represents the nominal phrase stem that took plural and/or possessive morphemes. NPS1 indicates that the nominal phrase is not completed yet. It may be marked with a case morpheme other than the nominative case. An example nominal phrase consisting of NS1, PLPMG and NPS1 constituents is given in (4.8) and the corresponding parse sub-tree is illustrated in Figure 4.8. In the example, we completed the phrase by the locative case mark. We explain the details of the case morpheme in the following parts of this section.

(4.8) kedi-ler-im-de

cat<NOM>-<Num:Pl>-<Poss:1s>-<Case:Loc>

'in my cats'

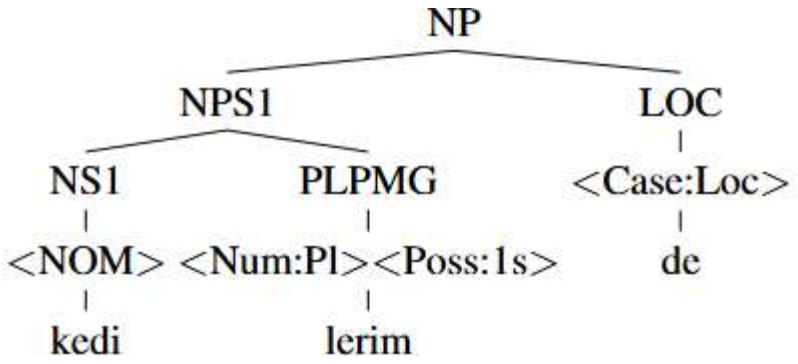


**Figure 4.8** The parse sub-tree of the phrase in (4.8).

When the nominal is marked with the nominative case and there are no further overt morphemes after the nominal inflection, PLPMG is located at the word boundary. In this case, we categorize the group as PLPMGB (plural and possessive morphemes group at the boundary). PLPMGB is combined with NS1 and completes the nominal phrase. An example phrase is given in (4.9) and the parse sub-tree of this phrase is shown in Figure 4.9.

(4.9) kedi-ler-im

cat<NOM>-<Num:Pl>-<Poss:1s><Case:Nom>

'my cats'

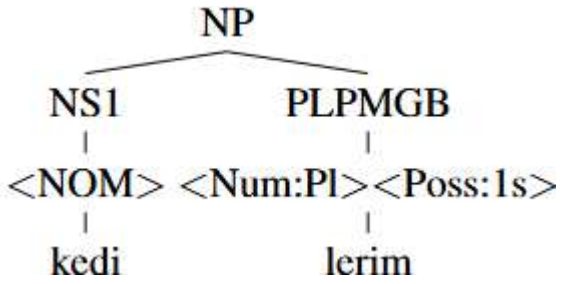


**Figure 4.9** The parse sub-tree of the phrase in (4.9).

A nominal phrase has different functions in the sentence, based on the case mark. Nominal phrases can be the subject or the object of a sentence, as well as nominal and verb qualifiers.

When a nominal phrase is marked with the nominative case, the phrase may function as a subject or object. Although the structure of a subject nominal phrase or object nominal phrase can be the same, we distinguish the constituents that show such phrases. We show a nominal phrase which functions as the object with symbol NP and a nominal phrase which functions as the subject with symbol NPSUB (nominal phrase as subject). We applied this discrimination because the behaviour of a verb phrase stem changes when a subject is attached to it. For example, such a verb phrase stem cannot take the infinitive suffix '-mak' or '-mek'. Example sentences where the noun 'kedi' is subject and object are given in (4.10-a) and (4.10-b) respectively. Corresponding parse trees are given in Figure 4.10.

(4.10) a. kedi gel-di
cat<NOM> come-<Tns:Past><Prsn:3s>
'cat came'

b. kedi sev-di-m
cat<NOM> pet-<Tns:Past>-<Prsn:1s>
'I petted cat'

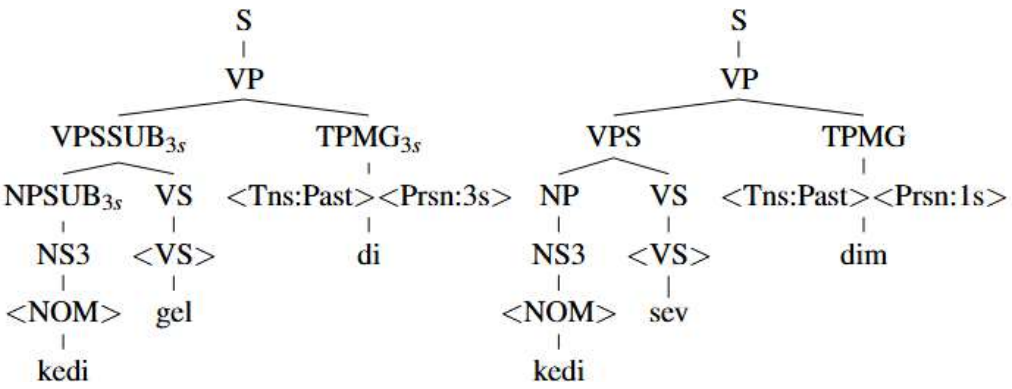


**Figure 4.10** Parse trees of the sentences in (4.10).

Having a subject in a sentence constrains an agreement rule. The subject and the verb must agree in terms of number and person. Therefore, further categorization is necessary for nominal and verbal constituents. The rules for the agreement are as follows,

- If the subject is first or second person, the verb must have first and second-person suffixes.
- If the subject is third person singular, the verb must have third person singular or to express respect[1], it must have third person plural suffixes.
- If the subject is third-person plural, the verb must have either third-person singular or plural suffixes.

Table 4.4 shows the classification and matching of the constituents based on the agreement.

**Table 4.4** Agreement of Subject and Verb

| **Subject** | **Agrees With** | |
|---|---|---|
| $NPSUB_{1S}$ | $TPMG_{1S}$ | |
| $NPSUB_{2S}$ | $TPMG_{2S}$ | |
| $NPSUB_{3S}$ | $TPMG_{3S}$ | $TPMG_{R}$ |
| $NPSUB_{1P}$ | $TPMG_{1P}$ | |
| $NPSUB_{2P}$ | $TPMG_{2P}$ | |
| $NPSUB_{3P}$ | $TPMG_{3S}$ | $TPMG_{3P}$ |

We treat the rest of the cases which have an overt form, differently. The differentiation is based on the behaviour and the function of the case-marked nominal phrase. We group the locative, ablative and instrumental cases under CMG (case morpheme group) and its boundary version CMGB (case morpheme group at the boundary) constituents. We show the dative, genitive and

[1] We use the subscript R to identify this exceptional case.

accusative cases with separate constituents because each one has different behaviours.

The nominal phrases which are marked with the locative, ablative or instrumental cases function as both verb qualifier (adverb) or noun qualifier. We show the nominal qualifier phrases by QP (qualifier phrase) constituent. Example sentences which contain a locative marked noun phrase are given in (4.11); parse trees which belong to the sentences are illustrated in Figure 4.11. In the first example, the nominal 'araba' matches with the NS2 constituent. This constituent shows the nominal stem which takes a case suffix. Here, we show the case mark with CMGB constituent because it is located at the word boundary, and it completes the phrase. NS2 is combined with CMGB constituent, and they form NPC (case marked nominal phrase). Since the case-marked noun phrase functions as a verb qualifier, it transforms into ADVP.

(4.11) a. araba-da uyu-du
Car<NOM>-<Case:Loc> sleep-<Tns:Past><Prsn:3s>
'slept in car'

b. çantada keklik
idiom
'fish in pan'

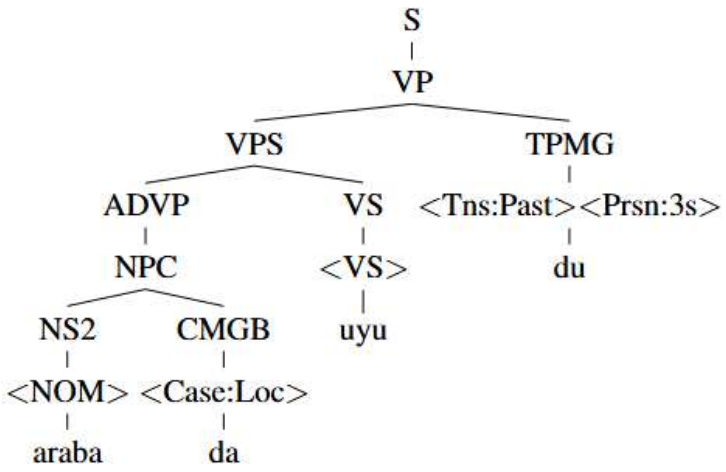


**Figure 4.11** The parse tree of the sentence in (4.11-a).

When there are further morphemes after the case, the case suffix is not located at the word boundary and cannot complete the noun phrase. In this situation, we use CMG to show the corresponding case morpheme. An example sentence is given in (4.12) and the corresponding parse tree is given in Figure 4.12. In this example, NS2 and CMG form the constituent NPS2 which shows a case marked nominal phrase stem. Next, a nominal deriving morpheme is attached to the nominal phrase stem NPS2 and from the nominal phrase which functions as the subject of the sentence. We use the relativizer suffix '-ki' in this example. We show nominal deriving morphemes with NDM (nominal deriving morpheme) or NDMB (nominal deriving morpheme at the boundary).

(4.12) araba-da-ki uyu-du
car<NOM>-<Case:Loc>-<Rel><NOM> sleep-<Tns:Past><Prsn:3s>
'one who is in car slept'

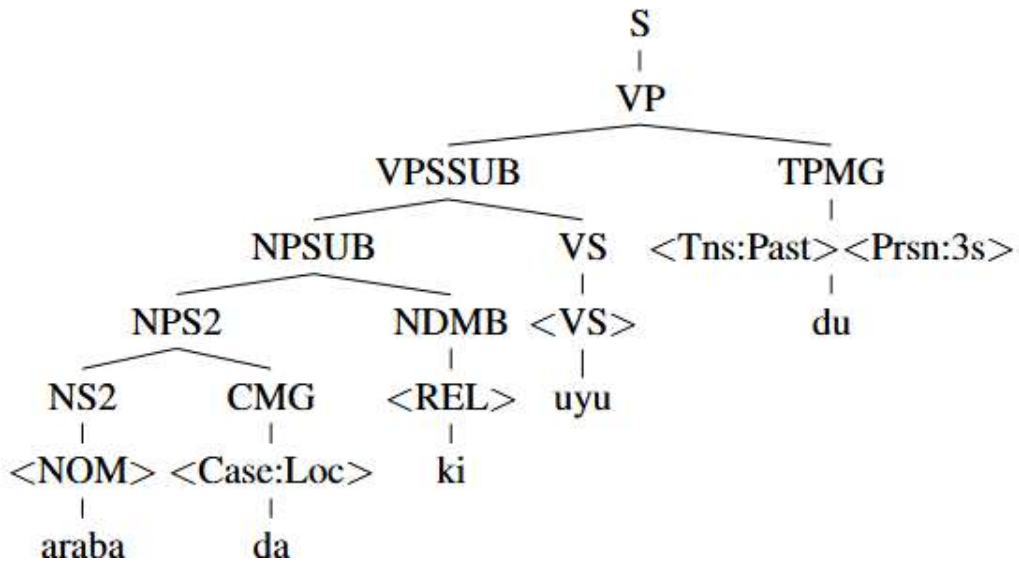


**Figure 4.12** The parse tree of the sentence in (4.12).

Noun phrases marked with the ablative case function as a verb or nominal qualifiers. Example sentences which have ablative marked nominal phrases are given in (4.13) and parse trees belonging to the sentences are illustrated in

Figure 4.13 and Figure 4.14. In the first example, the nominal phrase which is marked by the ablative case functions as a nominal qualifier. Therefore, the NPC constituent transforms into QP. We show the noun qualifiers with the QP constituent. Next, QP is attached to the NS3 which shows the nominal 'araba' and they form NP. The structure shown in the second example is like the locative case example.

(4.13) a. tahta-dan araba sür-dü-m
Wood<NOM>-<Case:Abl> car<NOM> drive-<Tns:Past>-<Prsn:1s>
'I drowe wooden car'

b. araba-dan in-di-m
car<NOM>-<Case:Abl> get out-<Tns:Past>-<Prsn:1s>
'I got out pf the car'

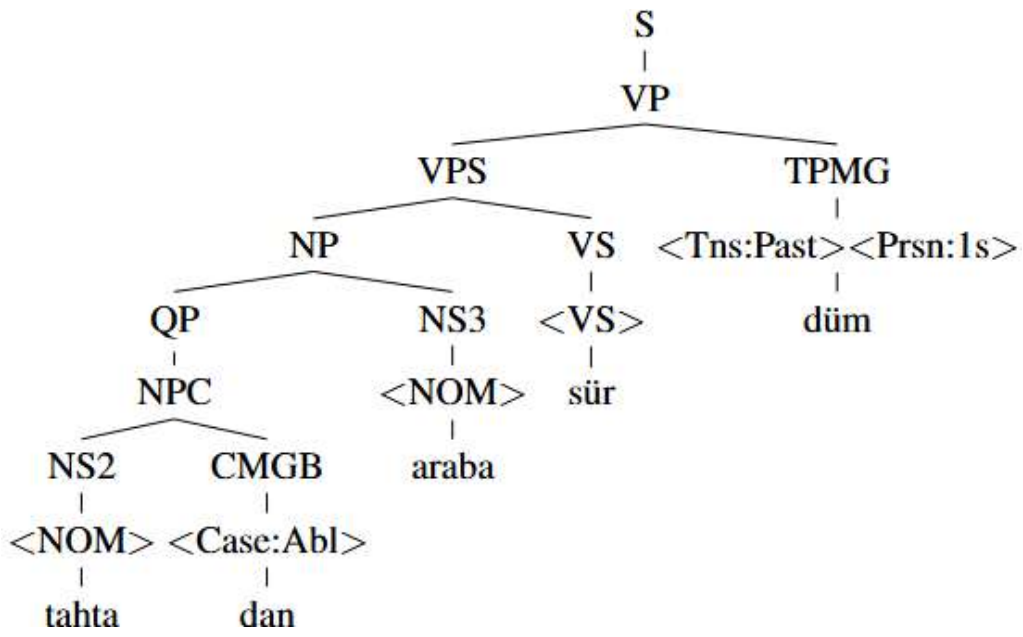


**Figure 4.13** The parse trees of the sentences in (4.13-a).

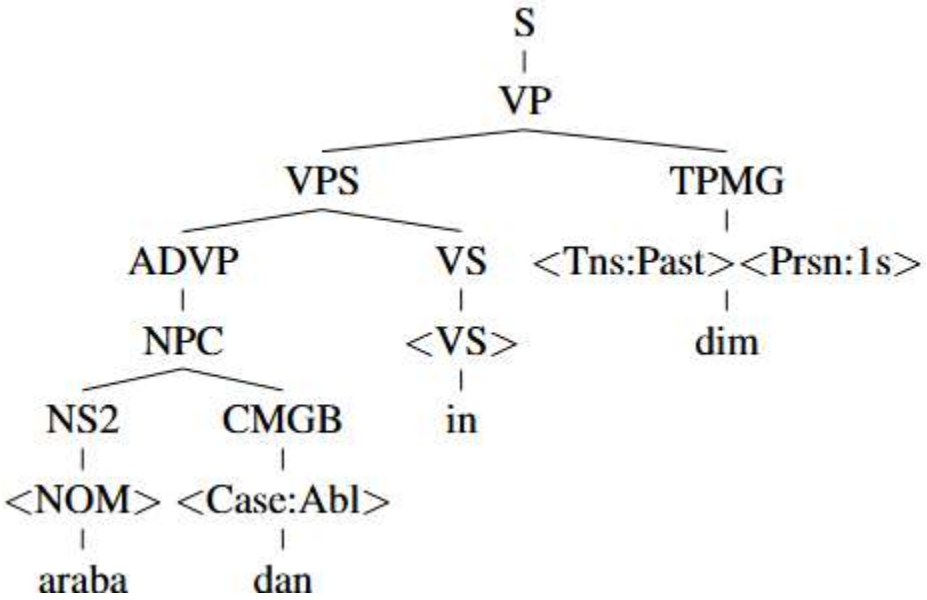


**Figure 4.14** The parse trees of the sentences in (4.13-b).

Instrument case makes the phrases which they attached to function as a verb or a nominal qualifier too. Example sentences in which instrument case marked nominal phrase functions as verb and noun qualifier are given in (4.14). The parse trees belonging to the sentences are shown in Figure 4.15 and Figure 4.16.

(4.14) a. araba-yla ev al-dı-m
car<NOM>-<Case:Ins> house<NOM> buy<VS>-<Tns:Past>-<Prsn:1s>
'I bought a car and house'

b. araba-yla gel-di-m
car<NOM>-<Case:Ins> come<VS>-<Tns:Past>-<Prsn:1s>
'I came by car'

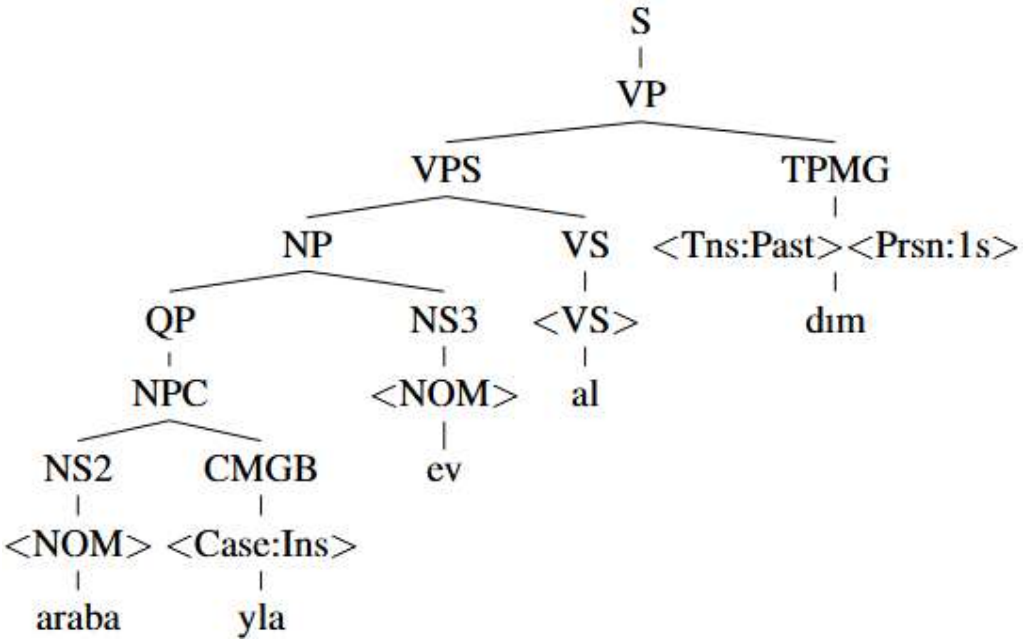


**Figure 4.15** The parse trees of the sentences in (4.14-a).

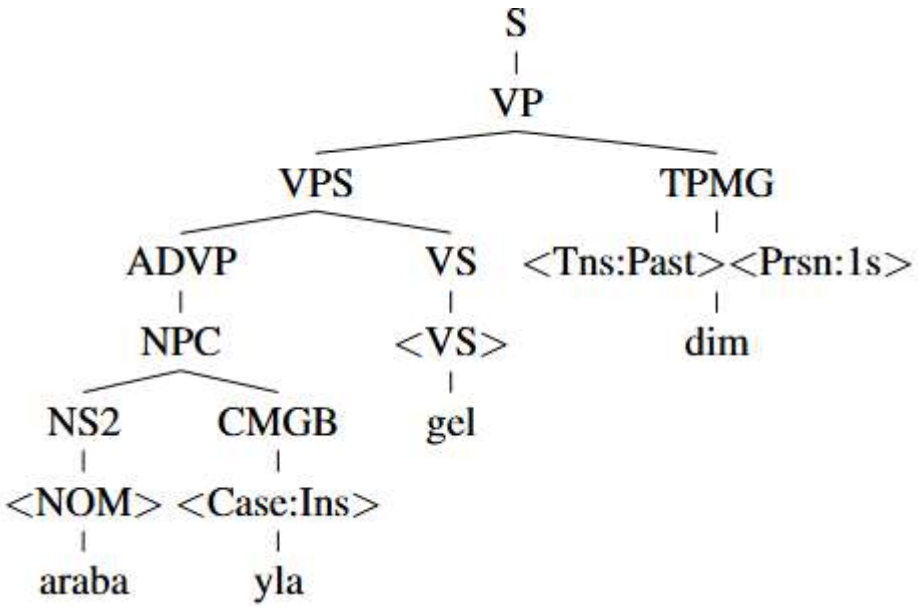


**Figure 4.16** The parse trees of the sentences in (4.14-b).

The Dative case-marked nominal phrases can only function as verb qualifiers. We show the dative case morpheme with DAT (dative) and dative

marked noun phrases as NPDAT (dative marked nominal phrase) constituents. An example sentence which contains a dative marked nominal phrase is given in (4.15) and the parse tree of the sentence is illustrated in Figure 4.17. The structure in the example is very similar to the verb qualifier structures of the phrases marked by locative, ablative and instrumental cases.

(4.15) araba-ya koş-tu
car<NOM>-<Case:Dat> run<VS>-<Tns:Past><Prsn:3s>
'He/She ran to the car'

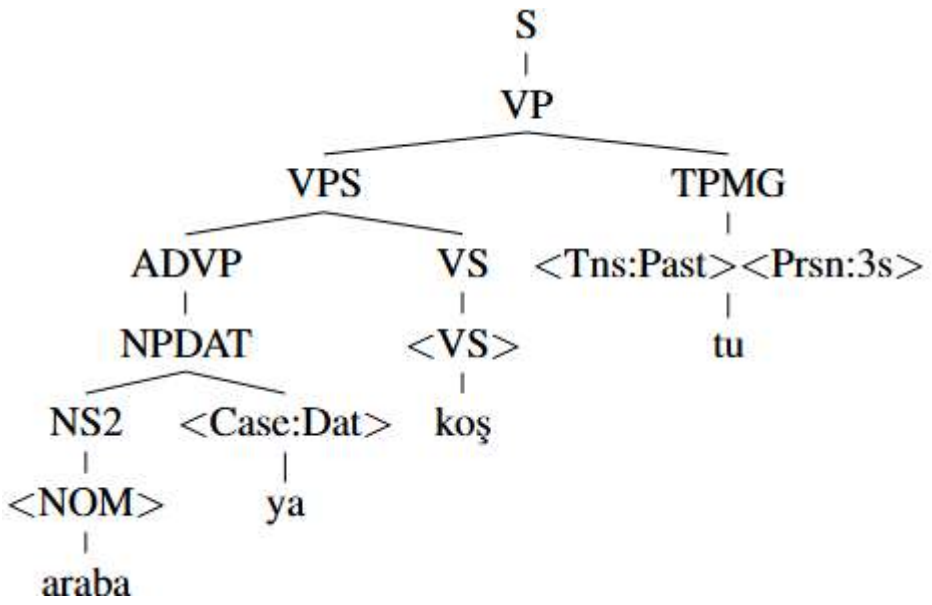


**Figure 4.17** The parse tree of the sentence in (4.15).

The actual reason why we separate the dative case from the others is, that the dative marked nominal phrases can be combined with postpositions to form adverb phrases. We show postpositions with POST (postposition) constituent. Postposition included version of (4.15) is given in (4.16). In the example, the dative marked noun phrase NPDAT is combined with the postposition 'doğru' to form the adverb phrase. The example parse tree is illustrated in Figure 4.18.

(4.16) araba-ya doğru<POSTP> koş-tu

car<NOM>-<Case:Dat> towards run<VS>-<Tns:Past><Prsn:3s>

'He/She ran towards the car'

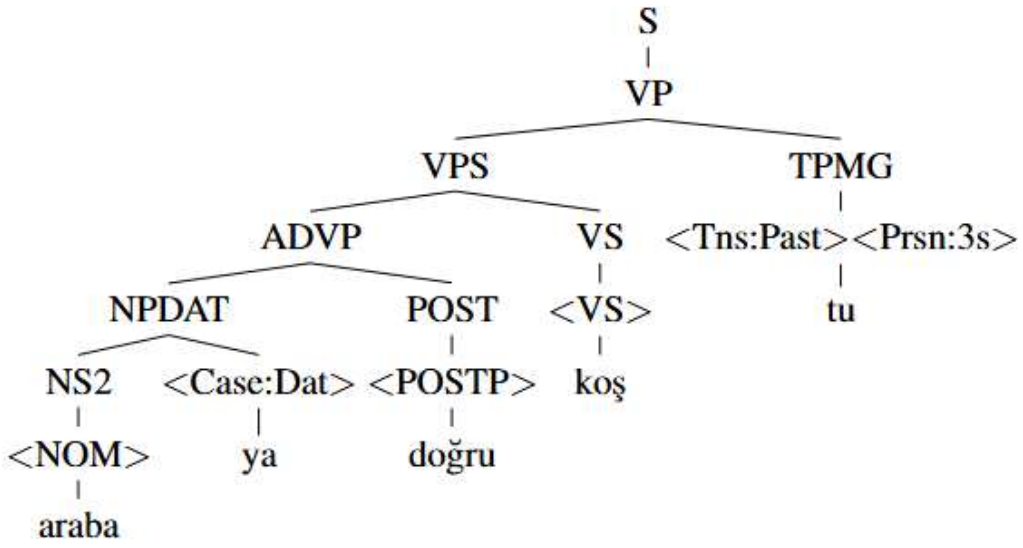


**Figure 4.18** The parse tree of the sentence in (4.16).

The genitive marked nominal phrases can only function as nominal qualifiers. The qualifier nominal which is formed by the genitive case has a unique behaviour as well. These phrases limit the possessive morpheme that the qualified noun can take. This means the qualifier nominal and the qualified nominal must agree in terms of person. Due to this behavioural difference, we show the genitive case morpheme with the GEN (genitive) constituent. We show the nominal phrases which are marked by genitive case with the NPGEN (genitive marked nominal phrase). To ensure the agreement between the qualified and the qualifier nominal, we categorize the fundamental constituents which appear in the genitive marked phrase structure based on the person. The categorizations of the relevant constituents are given in Table 4.5.

**Table 4.5** Categorization of NS1, NS2, NPS3, NPGEN, PLPLMG and PLPMGB constituents based on person categorization.

| **Person** | **Constituents** | | | | | |
|---|---|---|---|---|---|---|
| 1. Singular | $NS1_{1S}$ | $NS2_{1S}$ | $NPS3_{1S}$ | $NPGEN_{1S}$ | $PLPMG_{1S}$ | $PLPMGB_{1S}$ |
| 2. Singular | $NS1_{2S}$ | $NS2_{1S}$ | $NPS3_{2S}$ | $NPGEN_{2S}$ | $PLPMG_{2S}$ | $PLPMGB_{2S}$ |
| 3. Singular | $NS1_{3S}$ | $NS2_{3S}$ | $NPS3_{3S}$ | $NPGEN_{3S}$ | $PLPMG_{3S}$ | $PLPMGB_{3S}$ |
| 1. Plural | $NS1_{1P}$ | $NS2_{1P}$ | $NPS3_{1P}$ | $NPGEN_{1P}$ | $PLPMG_{1P}$ | $PLPMGB_{1P}$ |
| 2. Plural | $NS1_{2P}$ | $NS2_{2P}$ | $NPS3_{2P}$ | $NPGEN_{2P}$ | $PLPMG_{2P}$ | $PLPMGB_{2P}$ |
| 3. Plural | $NS1_{3P}$ | $NS2_{3P}$ | $NPS3_{3P}$ | $NPGEN_{3P}$ | $PLPMG_{3P}$ | $PLPMGB_{3P}$ |

An example sentence which contains genitive constructions is given in (4.17). In the example, the nominal ‘araba’ which is a third person singular nominal is shown with $NS2_{3S}$. This constituent is combined with genitive morpheme, and they form $NPGEN_{3S}$. This constituent shows a third person singular nominal which is marked by the genitive case. The genitive marked nominal ‘arabanın’ qualifies the nominal ‘teker’. They form the nominal phrase stem $NPS3_{3S}$. Lastly, the possessive morpheme of the same person is attached to the nominal phrase stem. The parse tree of (4.17) is illustrated in Figure 4.19.

(4.17) araba-nın teker-i patla-dı
car<NOM>-<Case:Gen> tire<NOM>-<Poss:3s> bust<VS>-<Tns:Past><Prsn:3s>
‘the tire of the car busted’

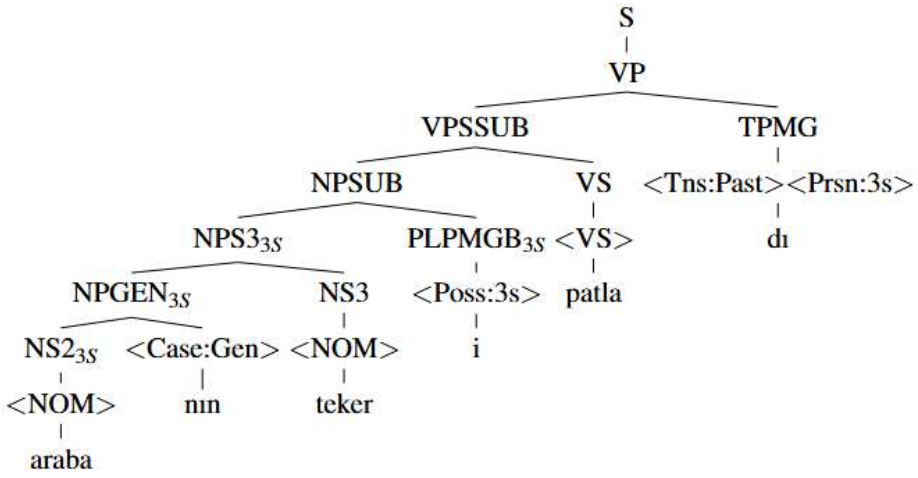


**Figure 4.19** The parse tree of the sentence in (4.17).

Accusative marked nominal phrases only function as the direct object of the sentence. We show the accusative case morpheme with ACC (accusative) and the accusative case marked nominal phrases with NPACC (accusative marked nominal phrase) constituents. An example sentence is given in (4.18). The parse tree of the sentence is given in Figure 4.20. In the example, the nominal 'araba' is shown with NS2 constituents because it waits for a case morpheme. NS2 and ACC constituent form NPACC constituent which represents the direct object of the sentence.

(4.18) araba-yı sürdü

Car<NOM>-<Case:Acc> drive<VS>-<Tns:Past><Prsn:3s>

'He/She drove the car'

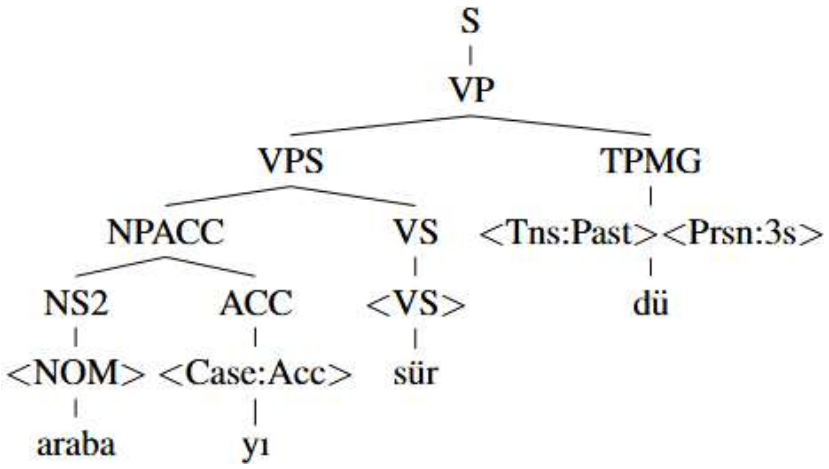


**Figure 4.20** The parse tree of the sentence in (4.18).

We demonstrated that in some cases morphemes make nominal phrases function as nominal qualifiers. In addition to nominals which become nominal qualifiers by the case marks, there are adjectives and nominal with qualification aspects. We show adjectives with the ADJ (adjective) and the nominals which have qualification aspect with $NS3_Q$ constituents. The qualifier phrase QP can be formed by ADJ or $NS3_Q$. Example sentences for both noun qualifications are given in (4.19) and corresponding parse trees are illustrated in Figure 4.21.

(4.19) a. kırmızı duvar gör-dü
red<NOM> wall<NOM> see<VS>-<Tns:Past><Prsn:3s>
'He/She saw red wall'

b. tuğla duvar gör-dü
brick<NOM> wall<NOM> see<VS>-<Tns:Past><Prsn:3s>
'He/She saw brick wall'

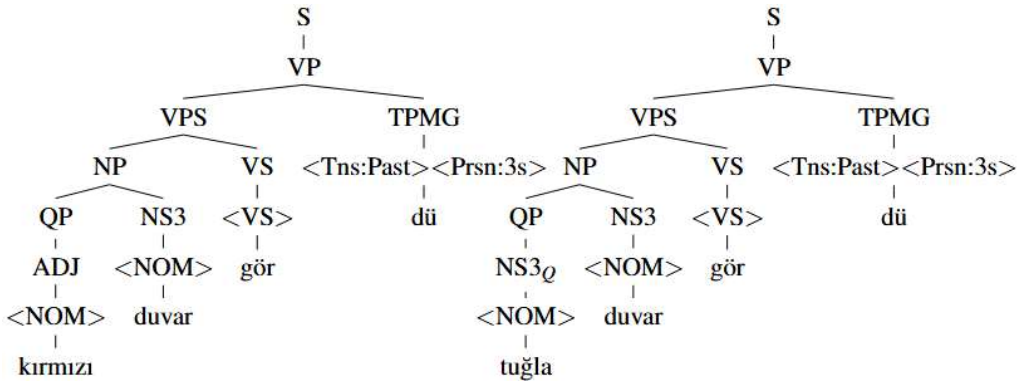


**Figure 4.21** The parse trees of the sentence in (4.19).

If the sentence does not contain a verb, it is called a nominal sentence. A nominal sentence requires a nominal predicate. We show nominal predicates with NPRED (nominal predicate). Any noun or adjective takes copula and person morphemes to form a nominal predicate. Next, the nominal predicate matches with a noun and forms the sentence. A nominal sentence example is given in (4.20). The parse tree of the sentence is illustrated in Figure 4.22.

(4.20) araba kırmızı-dır
car<NOM> red<NOM>-<Cop:Aor>
'the car is red'

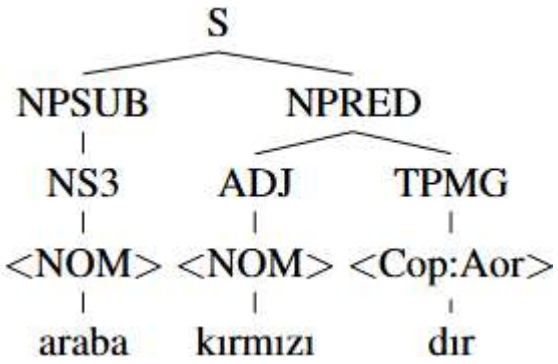


**Figure 4.22** The parse tree of the sentence in (4.20).

### 4.1.5. Adverb Constituents

Adverbs are the word types which qualify a verb. In addition to the words which are classified as adverbs, by an adverb deriving morpheme or a specific case morpheme, adverbs can be formed.

The most straightforward construction of an adverb phrase consists of only an adverb type of word. In this structure, the adverb is shown by the ADV constituent. ADV forms ADVP constituent alone. A sentence example where an adverb word forms the adverb phrase is given in (4.21). The parse tree which belongs to the sentence is shown in Figure 4.23. In the example, the adverb word 'çok' qualifies the verb 'uyu'.

(4.21) çok uyu-du

a lot<ADV> sleep<VS>-<Tns:Past><Prsn:3s>

'he/she slept a lot'

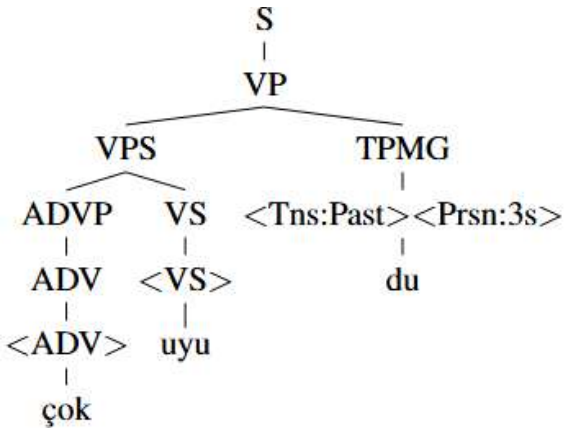


**Figure 4.23** Parse tree of the sentence in (4.21).

In Section 4.1.4, we explained how specific case marks form adverb phrases. Another way to form an adverb phrase is through an adverb-deriving morpheme. We show the adverb-deriving morphemes as the ADM (adverb-deriving morpheme) constituent. An example sentence where the adverb phrase is derived from a verb stem is given in (4.22). The corresponding parse tree is illustrated in Figure 4.24.

(4.22) otur-arak uyu-du
sit<VS>-<While> sleep<VS>-<Tns:Pst><Prsn:3s>
'he/she slept sitting'

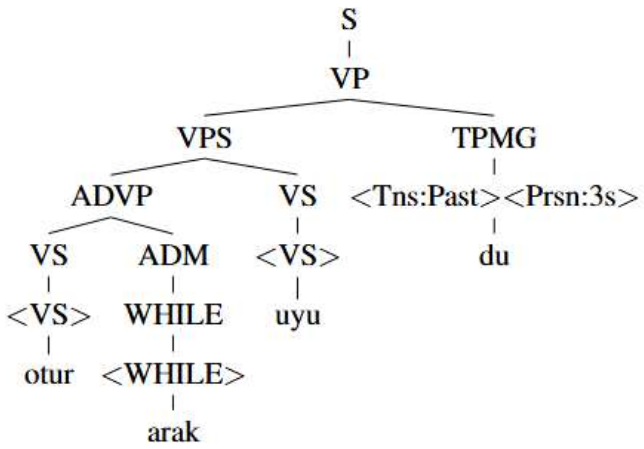


**Figure 4.24** Parse tree of the sentence in (4.22).

The word which forms an adverb phrase may be inflected before they are derived. In this structure, not only the word but the whole phrase stem structure which contains the qualifiers and the qualified stem form the adverb phrase. The example sentences which contain adverb phrases derived from noun and verb phrase stems are given in (4.23). The corresponding parse trees are illustrated in Figure 4.25 and Figure 4.26. In the first example, the adverb phrase is formed by the locative case marked noun phrase. The noun 'araba' is qualified by the adjective 'kırmızı', first. In the second example, the verb stem 'otur' is qualified by an adverb.

(4.23) a. kırmızı araba-da uyu-du

red<NOM> car<NOM>-<Case:Loc> sleep<VS>-<Tns:Pst><Prsn:3s>

'he/she slept in the red car'

b. yatak-ta otur-arak uyu-du

bed<NOM>-<Case:Loc> sit<VS>-<While> sleep<VS>-

<Tns:Pst><Prsn:3s>

'he/she slept sitting on the bed'

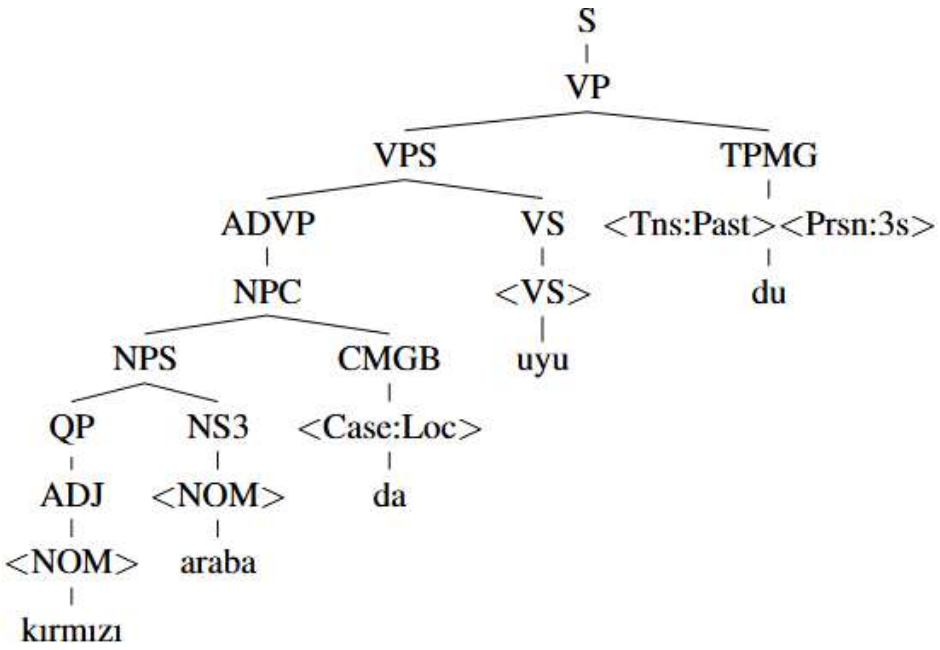


**Figure 4.25** Parse tree of the sentence in (4.23-a).

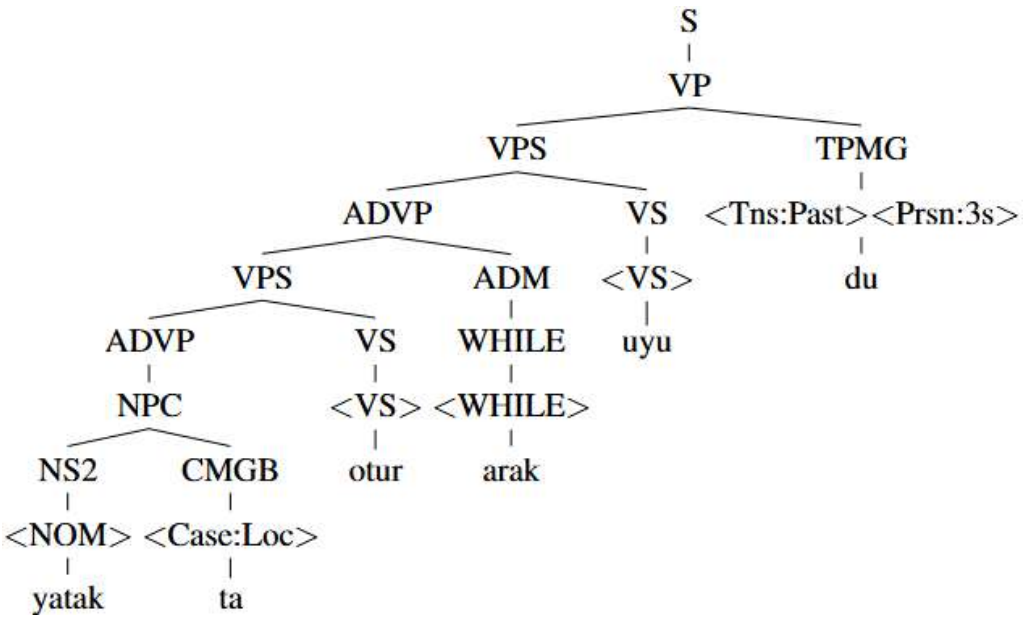


**Figure 4.26** Parse tree of the sentence in (4.23-b).

Any verb stem which has a subject can derive into an adverb phrase as well. While some of the adverb-deriving morphemes are compatible with the subject, some of them are not. For example, the '-arak' morpheme cannot derive a verb which has a subject. An example sentence is given in (4.2-a). To make a valid sentence, we need to use another adverb deriving morpheme such as in (4.24-b). We show the adverb deriving morphemes which are compatible with the subject by $ADM_{SUB}$ (adverb deriving morpheme compatible with the subject) constituent. There are sentences which are like the sentence in (4.24-a) yet valid. An example of such a sentence is given in (4.24-c). In the example, although the adverb is derived with a non-compatible morpheme, the structure and the sentence are valid because the subject belongs to the actual verb of the sentence, not the deriving verb. Parse trees of the valid sentences in (4.24) are illustrated in Figure 4.27 and Figure 4.28.

(4.24) a. *ben gel-erek sus-tu

I<NOM> come<VS>-<While> silence<VS>-<Tns:Past><Prsn:3s>

'I came in and shut up'

b. ben gel-ince sus-tu

I<NOM> come<VS>-<When> silence<VS>-<Tns:Past><Prsn:3s>

'he/she silenced when I came'

c. ben çalış-arak başar-dı-m

I<NOM> work<VS>-<While> succeed<VS>-<Tns:Past>-<Prsn:1s>

'I succeeded by working'

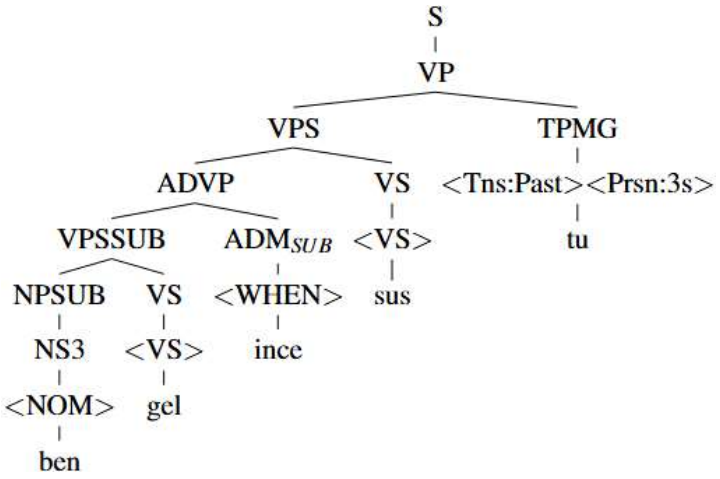


**Figure 4.27** Parse trees of the valid sentences in (4.24-b).

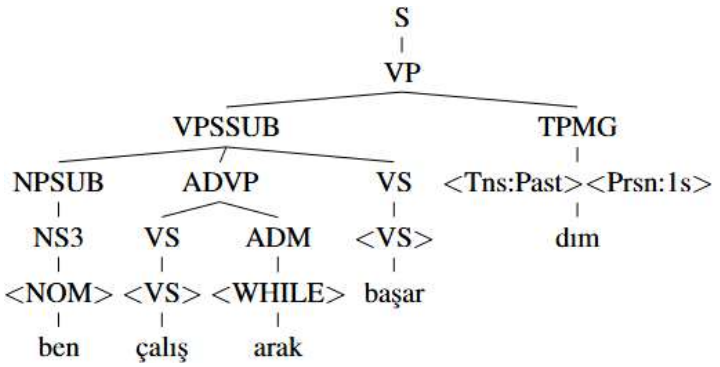


**Figure 4.28** Parse trees of the valid sentences in (4.24-c).

### 4.1.6. Postpositions

We show postpositions with the POSTP constituent in our grammar. They function in three different ways. Therefore, the constituent requires a categorization.

Postpositions produce either an adverb or qualifier phrase when they are combined with nominals. We show adverb producing postpositions as $POSTP_A$ and qualifier producing ones as $POSTP_Q$ Examples are given in (4.25) and (4.26) respectively. The parse trees of the examples are illustrated in Figure 4.29 and Figure 4.30.

(4.25) kedi gibi zıpla-dı
cat<NOM> like<POSTP> jump<VS>-<Tns:Pst><Prsn:3s>
'he/she jumped like a cat'

(4.26) kedi gibi adam zıpla-dı
cat<NOM> like<POSTP> man<NOM> jump<VS>-<Tns:Pst><Prsn:3s>
'the man like a cat jumped'

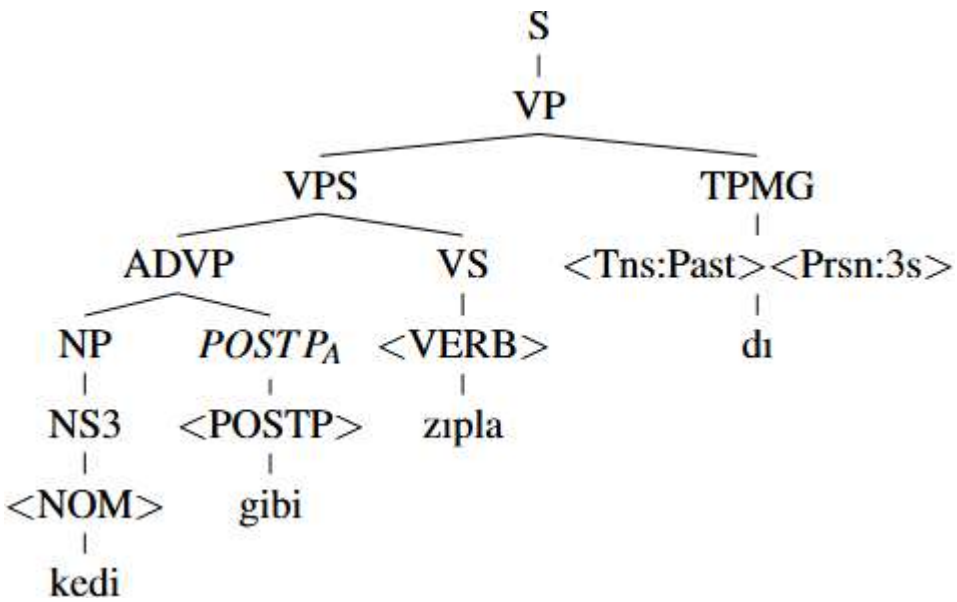


**Figure 4.29** Parse tree of the sentence in (4.25).

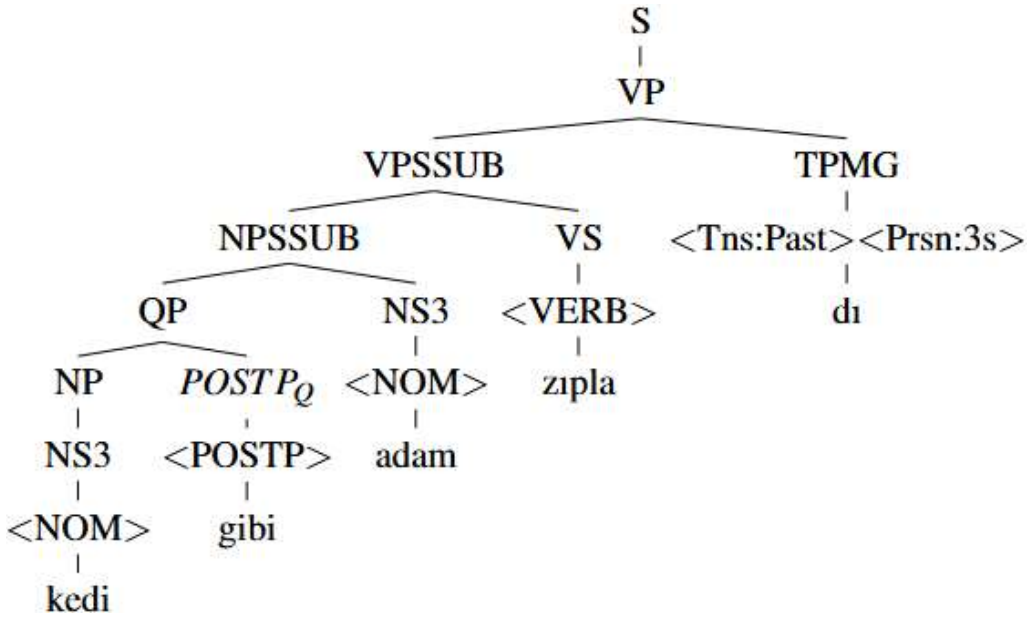


**Figure 4.30** Parse tree of the sentence in (4.26).

The third function of postpositions is repeating the type of the phrase that it is combined with only once. We name this type of postposition as inclusive and show them with $POSTP_i$. Additionally, to prevent further repetitions, we mark the resulting constituent with the i subscript as well. An example of such a postposition is given in (4.27) and the relevant parse tree is illustrated in Figure 4.31. In the tree, the postposition "bile" repeats the NPACC, and the result appears as $NPACC_i$.

(4.27) cocuğ-u bile iste-di
child<NOM>-<Case:Acc> even<POSTP> want<VS>-<Tns:Pst><Prsn:3s>
'he/she even wanted the child'

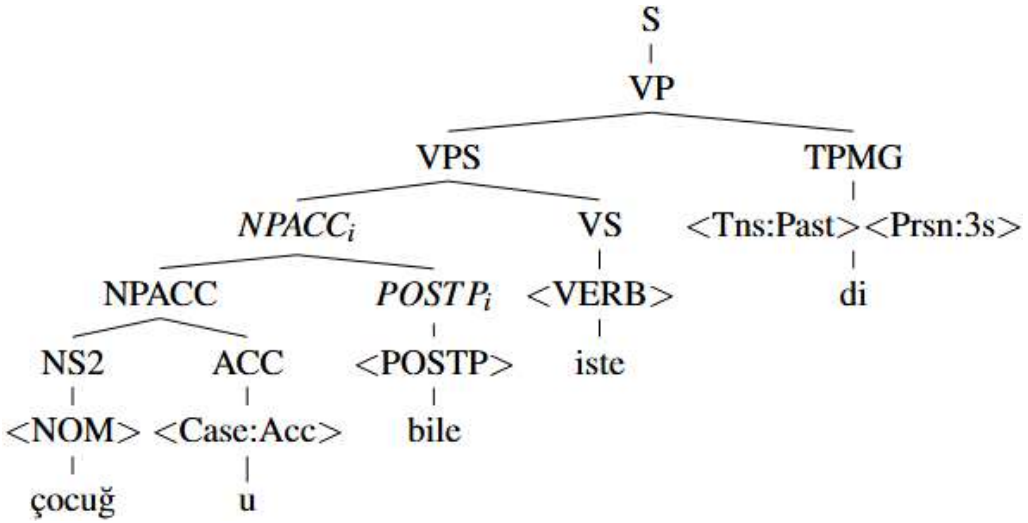


**Figure 4.31** Parse tree of the sentence in (4.27).

There is one exceptional postposition which can be combined with a repeated phrase. We show such a postposition with the subscript q. An example sentence is given in (4.28) and the corresponding parse tree is illustrated in Figure 4.32.

(4.28) cocuğ-u bile mi iste-di
child<NOM>-<Case:Acc> even<POSTP> want<VS>-<Tns:Pst><Prsn:3s>
'did he/she even want the child'

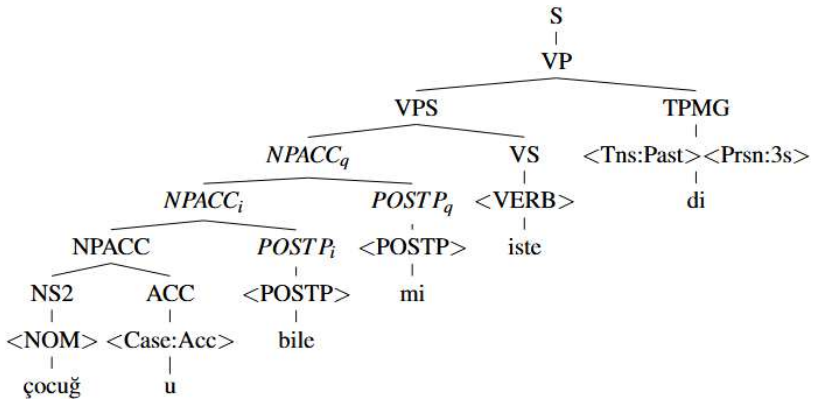


**Figure 4.32** Parse tree of the sentence in (4.28).

### 4.1.7. Word Order

Turkish has a relatively free word order. This means the order of the words in a sentence can change and yet, the sentence is valid with the new order of the words. Since the constituency grammar rules are defined based on the order of the constituents, we include constituents, rules and structures to recognize different orders.

In fact, the problem with the word order starts when the syntactic elements come after the verb. The stemmed phrase structure that we use is formed when the phrases are combined with a root or stem. Then, the stemmed phrase is combined with a further inflection group, and they form a phrase. This structure forces the elements to come before the verb. The last grow which completes the sentence will be the attachment of tense and person suffixes to the structure. It is not possible to apply this idea when something comes after the verb. Practically, it requires omitting the tense and person which is in fact impossible to do since the rules structure does not allow that.

In such a case, we use an incomplete phrase constituent. The incomplete phrase is formed in the usual way. For example, if it is an incomplete verb phrase, first, the verb root or stem and the qualifiers which come before it in the sentence order are combined and form a stemmed verb phrase. Then, again, as usual, the stemmed verb phrase is combined with tense and person group, and they form an incomplete verb phase. We show such a phrase with $VP_M$ constituent where the subscript M denotes the verb phrase requires to be completed. Two different order variations of 'köpek adamı ısırdı' (the dog bit the man) are given in (4.29) and the respective parse trees are illustrated in Figure 4.33 and Figure 4.34.

(4.29) a. köpek ısır-dı adam-ı

dog<NOM> bite<VS>-<Tns:Past><Prsn:3s> man<NOM>-<Case:Acc>

'The dog bit the man'

b. ısır-dı köpek adam-ı

bite<VS>-<Tns:Past><Prsn:3s> dog<NOM> man<NOM>-<Case:Acc>

'The dog bit the man'

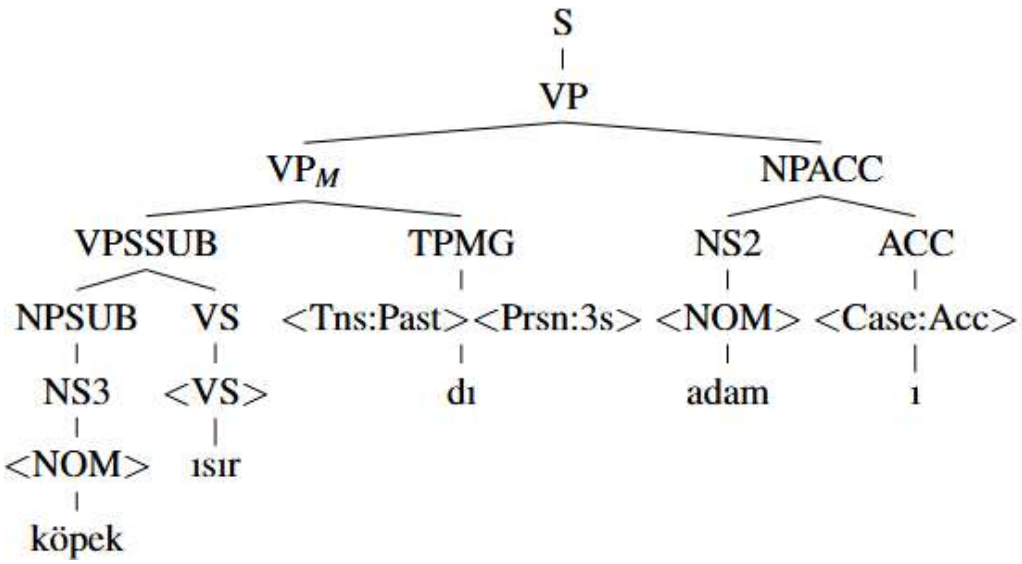


**Figure 4.33** Parse tree of the sentence in (4.29-a).

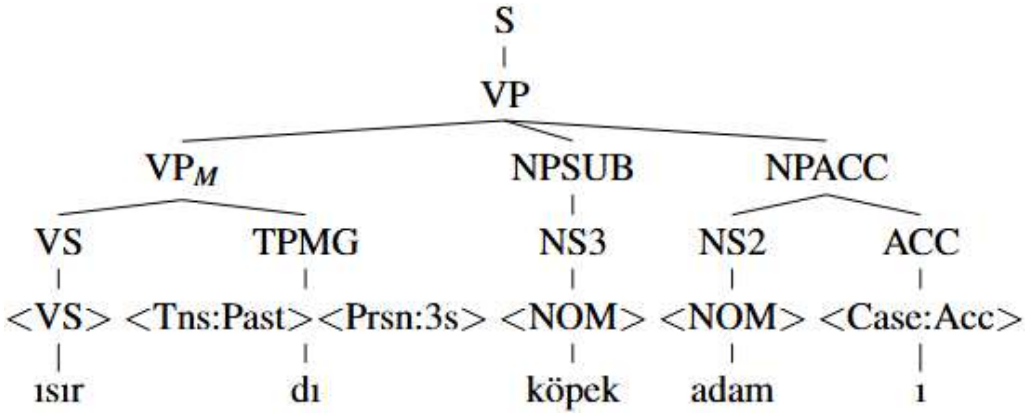


**Figure 4.34** Parse tree of the sentence in (4.29-b).

To handle the changing word order, the grammar must have the rules which express each order. This situation becomes a permutation problem when the rule which expresses the dominant word order exists.

The rule which expresses the order of the sentence 'köpek adamı ısırdı' is given in (4.30-a). The rules which express the orders of the sentences in (4.29) are given in (4.30-b). Also, the rules which express the rest of the permutations of the dominant word order are given in (4.30-c).

(4.30) a. VPSSUB→NPSUB NPACC VS
VP → VPSSUB TPMG
b. VP → $VP_M$ NPACC
VP → $VP_M$ NPSUB NPACC
c. VP → $VP_M$ NPSUB
VP → $VP_M$ NPACC NPSUB
VPSSUB→NPACC NPSUB VS

In (4.30-a), the rules show that the attachments of the qualifiers to the verb stem. In (4.30-b), rules show the order when either the object or both the subject and the object are placed after the verb. The rules in (4.30-c) indicate three different orders. The first shows that only the subject is places after the verb. The object is included in $VP_M$ and the subject is combined with the incomplete

verb phrase. The second one shows that both the subject and the object are places after the verb. But in this case, the object comes before the verb. Since the production rules are fully order-sensitive, this situation brings a new rule in addition to the one in (4.30-b). $VP_M$ includes only the verb stem and TMPG. All qualifiers combined with the incomplete verb phrase. And lastly the third one shows the order where the subject and the object comes before the verb yet, unlike the one in (4.30-a), the object comes before the subject. Nevertheless, there is no incomplete verb phrase. All qualifiers are combined with the verb stem.

## 4.2 SENTENCE ANNOTATION TOOL FOR MORPHOSYNTACTIC GRAMMAR

Building a treebank is helpful in order to obtain a constituency grammar. It involves manually annotating sentences. Since the annotation is a hard process, we implemented a tool to ease this process.

SynEdit is a visual tree annotation tool which allows you to use select morphological analysis of each token, group morphemes to create leaves and connect nodes to build the syntax tree, (Özenç & Solak, 2019). We developed SynEdit as a desktop application using the PyQt5 library of Python with Model View Controller architecture.

SynEdit allows you to keep all your tree files in a folder and it treats the folder as a treebank project. It provides a default set of constituents for each treebank individually. The constituent set can be modified anytime during the treebank construction. If the modification of the set affects any annotated sentences, the application lets you know. Similarly, you can track which sentences are fully, partially or not annotated. The application stores annotations in Penn-Treebank-like (Marcus et al., 1993) format and it allows exporting images for annotated sentences.

SynEdit recognizes leaf and non-leaf nodes, and it shows you only the suitable constituent to be used in the node during the sentence annotation. In

addition, it can recognize any multi-word expressions based on the lexicon of the morphological analyser and it allows you to merge the separate tokens to form a single token for the multi-word expression if necessary.

The tool can generate a constituency grammar based on all fully annotated sentences in two different ways: normal and enhanced. In the normal generation, all the rules extracted from the valid parse trees are dumped into a file as a set of unique productions. In the enhanced generation, in addition to the rules extracted from the valid parse trees, it generates unary rules related to the roots, morphemes and morpheme groups from the structure of the morphological analyser. The grammar can recognize more terminal symbols even though they do not appear in the parse trees.

The interface of the tool is shown in Figure 4.35. We plan to add more user-friendly features during the future improvements of the project and the tool.

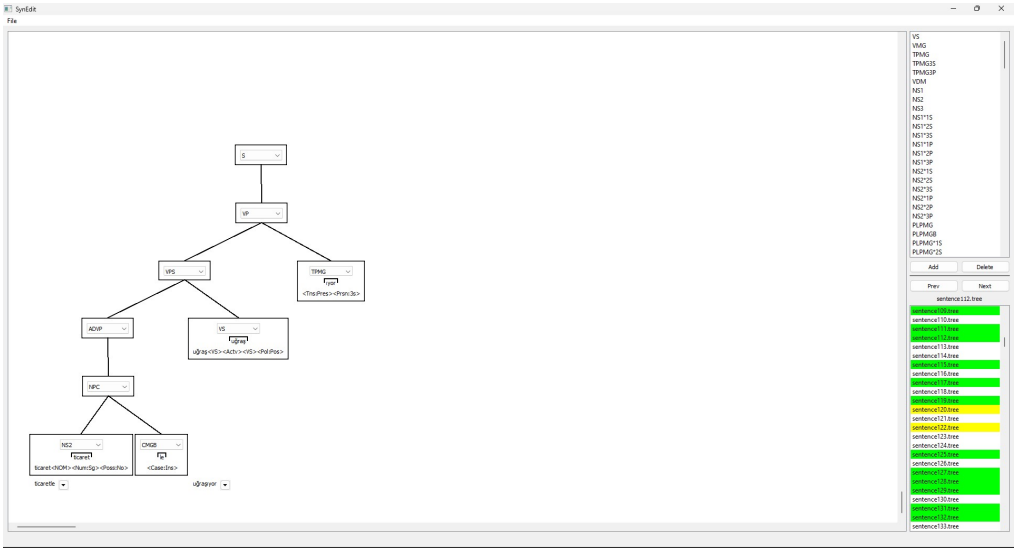

**Figure 4.35** SynEdit Interface.

## 4.3 TREEBANK AND GRAMMAR

We constructed a mini treebank[2] of 150 sentences to demonstrate our study. Our manually constructed treebank has some sentences from TrMor2018 (Dayanik et al., 2018) which is a dataset of morphologically annotated sentences. We randomly picked 1000 sentences which are shorter than 7 words. We cleared punctuation in sentences. We want to keep the morphological operations limited to the common nouns. Therefore, we neglected the sentences with proper nouns.

We did the annotations on SynEdit. During the annotations, no third parties are involved in the process. Although SynEdit can yield a probabilistic context-free grammar by calculating probabilities for each rule, we do not use probabilities due to the small number of annotated trees.

On the same annotated sentences, we can generate two different CFGs as the standard and enhanced. The standard one consists of 945 rules retrieved from the annotated trees. Several statistics related to the grammar are given in Table 4.6.

**Table 4.6** Counts of different rule types in the grammar

| Rule Type | Count |
|---|---|
| Terminal | 679 |
| Non-terminal | 266 |
| Unitary Terminal | 673 |
| Unitary non-terminal | 6 |
| Binary | 92 |
| Ternary | 61 |

[2] https://github.com/MorphosyntacticConstituencyTreebank/MiniTreebank

The enhancement we do on grammar increases the number of rules in grammar by 99347. Newly added rules consist of only unitary rules which add new terminal symbols to the grammar.

We are aware that, 150 sentences are far from providing comprehensive grammar and building a statistics-based grammar. Nevertheless, it allows us to illustrate our approach to parsing and grammar.

# CHAPTER 5

## 5. MORPHOSYNTACTIC CONSTITUENCY PARSING

Parsing is an operation to solve membership problems in the linguistic context. Precisely, it is to query whether a sentence belongs to a particular language and if it is, construct its parse tree. Such a process requires grammar which represents the model of the language. In constituency parsing, the same objective is focused on the constraint of grammar to be a constituency grammar.

There are various rule-based algorithms and implementations of constituency parsing. In this dissertation, we focus on the CYK algorithm.

The traditional CYK algorithm works on the token level. First, it replaces the tokens with constituents then recursively combines constituents based on the grammar rules until the top constituent S is produced. The CYK algorithm recognizes a sentence as a member of a language.

We explained the CYK algorithm's process in detail and the inadequacies when it is used for a language such as Turkish in Chapter 3. In this chapter, we share the details of our studies to involve the morpheme-based grammar of Turkish in the CYK process.

### 5.1 EXTENDED CYK ALGORITHM

Considering the Turkish, the main obstacle is the lowest level of the algorithm. Turkish grammar requires morphemes to be included, but the standard algorithm cannot reach them. We extended the CYK algorithm to start parsing on the morpheme level.

Our extension to the CYK algorithms consists of two major algorithms: preprocessing and parsing. While preprocessing handles tasks such as input preparation and generation of features necessary for parsing, parsing handles the rest. In the following sections, we share the details of the preprocessing and

additions which we do in the traditional CYK parsing. But before any specific operations, the grammar-related tasks must be done as in any CYK implementations. The Grammar is read as input and stored in a dictionary structure. Each rule in the grammar is stored as key-value pairs where the right-hand side (RHS) is the key, and the left-hand side (LHS) is the value.

The CYK algorithms constraints the CFG used, to be in CNF. Therefore, we apply the standard CNF conversion to the input grammar. The CNF grammar is stored in a separate dictionary in the same form.

We will follow up the example sentence 'dişi oyan bilir' to explain the details of preprocessing and parsing step by step. This sentence has two unique readings in terms of syntactic analysis. The readings are given in (5.1). Moreover, it allows us to illustrate each detail in the system. Zero morphemes are omitted to simplify the example.

(5.1) a. diş-i oy-an bil-ir
tooth<NOM>-<Case:Acc> drill<VS>-<Agt:Aor><NOM> know<NOM>-<Tns:Aor>
'the one who drills the tooth knows'

b. dişi oya-n bil-ir
female<NOM> embroidery<NOM>-<Poss:2sg> know<VS>-<Tns:Aor>
'your female embroidery knows'

### 5.1.1. Preprocessing

Preprocessing starts when the sentence to be analysed is taken in a list of tokens form. Although this sentence is the input of our extended parser, it is not used in the parsing stage. Preprocessing consists of 4 steps. Each step includes different operations to prepare the parsing input, and features used in the parsing stage. The first step of preprocessing is feature generation.

The features generated are mostly based on morphological forms and information. Therefore, each token of the sentence is put through the

morphological analysis first. We use MorTur (Özenç & Solak, 2020) as the analyser. All morphological analyses of the tokens in the example sentence are given in (5.2). The morphological analysis is conducted on each token individually, regardless of the context. It is possible to do a disambiguation to eliminate the irrelevant analyses. But we do not disambiguate on purpose since we want to get all the possible parses.

(5.2) a. [[‘<NOM><Num:Sg>’, ‘<NC><Case:Nom>’], [‘diş’, ‘i’]],
[[‘<NOM>’, ‘<Num:Sg><Poss:3s><Case:Nom>’], [‘diş’, ‘i’]],
[[‘<NOM><Num:Sg><Poss:No>’, ‘<Case:Acc>’], [‘diş’, ‘i’]],
[[‘<NOM><Num:Sg><Poss:No><Case:Nom>’], [‘dişi’]]

b. [[‘<VS><Actv><VS><Pol:Pos>’,
‘<AgtA><NOM><Num:Sg><Poss:No><Case:Nom>’], [‘oy’, ‘an’]],
[[‘<NOM>’, ‘<Num:Sg><Poss:2s><Case:Nom>’], [‘oya’, ‘n’]]

c. [[‘<VS><Actv><VS><Pol:Pos>’,
‘<AgtA><NOM><Num:Sg><Poss:No><Case:Nom>’], [‘bil’, ‘ir’]],
[[‘<VS><Actv><VS><Pol:Pos>’, ‘<Tns:Aor><Prsn:3s>’], [‘bil’, ‘ir’]],
[[‘<NOM><Num:Sg><Poss:No><Case:Nom>’], [‘bilir’]]

Each analysis in (5.2) is a pair of morpheme lists. Where the first one is the list of abstract morphemes, and the second one is the list of surface morphemes. Abstract morphemes are grouped as synchronised to the morpheme group constituents we have. A summary of the constituents is given in 0. In (5.2), the abstract morphemes which have zero surface form are grouped.

A list of features is prepared for each token. In the list, each token has its feature dictionary which contains, the token, the analyses, the most split form, possible roots, and possible morphemes of the token. A dictionary is filled if the token has at least one morphological analysis. The corresponding algorithm is given in Algorithm 5.1.

**Algorithm 5.1 Feature Generation**

***Input:*** *sentence, analysedSentence*
***Output:*** *features*

1 *Let* ***features*** *be a list of dictionaries*
2 ***For*** *each* ***token*** *in sentence*
3 *Let* ***tokenFeatures*** *be a dictionary*
4 *Insert all analyses of the* ***token*** *into* ***tokenFeatures***
5 *Compute most split and insert into* ***tokenFeatures***
6 *Insert all roots of the token into* ***tokenFeatures***
7 *Insert all affixes of the token into* ***tokenFeatures***
8 *Insert* ***tokenFeatures into*** *features*
9 ***End***

The most split is the most fragmented form of the token. It is a list of fragments where each fragment can be the root, any morphemes or units smaller than morphemes such as characters. We use most split of each token to form the input of the parsing phase. Also, each fragment is used to construct bigger morphemes during the parsing process. Most splits of the tokens 'dişi', 'oyan' and 'bilir' are given in (5.2).

(5.2) a. ['diş', 'i']
b. ['oy', 'a', 'n']
c. ['bil', 'ir']

We calculate the most split for each token, based on the analysis which has the largest number of surface morphemes. The morphemes of all other analyses are compared with the morphemes of the longest one. Any morpheme which has an exact match is transferred into the most split as a whole. If matching between the morphemes is only on some specific characters, not the whole morphemes but the matching characters inserted into the most split (e. g. 'a' and 'n' in (5.2-a)).

Next, all possible roots of each token are collected in Possible Roots. We scan all the analyses of each token and collect all unique roots in the designated lists of token feature dictionaries. This list is used in Root Rule generation later. The lists of Possible Roots for each token in the example are given in (5.3).

(5.3) a. [‘diş’, ‘dişi’]

b. [‘oy’, ‘oya’]

c. [‘bil’, ‘bilir’]

In addition to all possible roots, all suffixes that the token has are also processed. In a similar way, we scan all the analyses of the token and store all the suffixes that exist. The surface form of a suffix may have multiple abstract morphemes. Therefore, we store the suffixes in a designated dictionary where the unique surface is a key and the list of all abstract morphemes of that form is the value. The surface forms are used in the Root Rule generation and the corresponding lists of abstract morphemes are used in the filter generation later. The key-value pairs of Morphemes for each token in the example are given in (5.4).

(5.4) a. ‘i’: [‘<NC><Case:Nom>’, ‘<Num:Sg><Poss:3s><Case:Nom>’, ‘<Case:Acc>’]

b. ‘an’: [‘<AgtA><NOM><Num:Sg><Poss:No><Case:Nom>’],

‘n’: [‘<Num:Sg><Poss:2s><Case:Nom>’]

c. ‘ir’: [‘<AgtA><NOM><Num:Sg><Poss:No><Case:Nom>’, ‘<Tns:Aor><Prsn:3s>’]

The input of the parsing phase, abstract morpheme filters and root rules are produced after all features are generated.

The parsing input is a list of morphemes or sub-morphemic structures although the input of the overall system is a list of tokens. We acquire such a list by collecting all the elements in each most split in one list. Such a list which is formed for the example input is given in (5.5).

(5.5) [‘diş’, ‘i’, ‘oy’, ‘a’, ‘n’, ‘bil’, ‘ir’]

The parsing input generation is followed by the filter generation. We use filtering to prevent wrong constituents from being added to the table.

The traditional CYK parsing can yield all the parses of a given list of tokens. It enables us to have such a feature since all different POS of each token are the words only.

On the morpheme level, the token can be an unaffixed word, stem or an affix. Such diversification in tokens causes the appearance of wrong constituents due to homonymous tokens in the parsing input. Particularly, this is an issue when the homonym of the token belongs to different types such as an affix and a word. For example, ‘an’ in (5.1-a). Although ‘an’ is not in the parsing input given in (5.5) we involve it in additional steps. We will detail that process in the next step of the preprocessing. For the moment, let’s just focus on filtering and assume that ‘an’ is already involved.

‘an’ is both a word (nominal, *moment*) and an affix (nominal deriving affix) on the morpheme level. A comprehensive morphosyntactic grammar should have both types of ‘an’ but in the parsing table, we can have only one of them. Of course, it is required to find the one which is indicated by the morphological analyses and the abstract morphemes. In this example, although ‘an’ has two different abstract morphemes ‘<NOM>’ and ‘<AGTA>’, the first one is eliminated by the filters of ‘an’. We omit abstract morphemes that may appear due to affixes with zero surface form to simplify the example. The actual abstract morpheme of the affix ‘an’ is in (5.4-b) for this example.

We generate a filter for each morphological element of each token. The process here can be expressed as a conversion from an abstract morpheme to a constituent through the CFG. Based on the unary rules that involve the abstract morpheme, we find all top-level constituents. Thus, we can filter the constituents that appear during paring. The CFG rules related to both homonyms of ‘an’ are given in (5.6). Filters of ‘an’ are generated regarding the rules in (5.6-a).

(5.6) a. NDMB→<AGTA>

NDM→<AGTA>

<AGTA>→an

b. NPSUB → NS3

NP→NS3

NS3→<NOM>

<NOM>→an

Next, the root rules are generated. These rules are temporary rules which are specifically generated for each input. The sole purpose of the root rules is to generate all roots, morphemes, morpheme groups or multiword expressions during the parsing process in the given input. Thus, we enable our system to involve each various morphologic part of a word in parsing from a single input such as given in (5.5).

We design the root rules in the form of CNF grammar rules, because we want them to function in the natural process of CYK. Yet, we never consider them as actual grammar rules and we do not store them in the same rule dictionary. As in CNF, there are two types of root rules: unary and binary. The unary root rules convert a surface element[3] into a special form which is not like any constituent (e.g. from 'diş' to '{diş}'). On the other hand, binary root rules function as morphotactic combinations. They combine two special forms into a new one which is particularly present in one of the analyses of the corresponding token.

We generate root rules based on the elements in the Most Split, Possible Roots and Suffixes features. Recursively, consecutive elements in the Most Split are combined in pairs. If the result of the combination is in either Possible Root or Suffixes, two unary root rules for individual elements which are combined and a binary rule which represents the combination are generated. Otherwise, although the combination result does not exist, it may produce an existing form through further combination. This means, that to obtain a certain surface form,

[3] Here, surface element means any element in the parsing input.

it may require employing several consecutive combinations. This shows the necessity of multiple binary root rules. Therefore, we check the next elements in the token's Most Split whether they yield an existing form because of the combination with the non-existing one. If such a situation occurs, all necessary root rules are generated and some of them are used as an intermediate step. Any valid elements acquired during root rule generation go through the filter generation process.

Let's focus only on the token 'oyan' in the example to examine the root rule generation process. Most Split for 'oyan' is ['oy', 'a', 'n']. The recursive combination of consecutive elements yields three new forms as 'oya' from 'oy'+'a', 'oyan' from 'oya'+'n' and 'an' from 'a'+'n'. Since only the two of them are valid in terms of the morphological elements belonging to the token, corresponding root rules are generated. Such root rules are given in (5.7-b). The root rules generated for the input 'dişi oyan bilir' are given in (5.7). The root rule generation algorithm is given in Algortihm 5.2.

(5.7) a. {diş}→diş
{i}→I
{dişi}→{diş}{i}
b. {oy}→oy
{a}→a, {n}→n
{oya}→{oy}{a}
{an}→{a}{n}
c. {bil}→bil
{ir}→ir
{bilir}→{bil}{ir}

**Algorithm 5.2 Root Rule Generation**

***Input:*** *features*
***Output:*** *dictionary of root rules*

```
1   Let rootRules be a dictionary
2   For each f in features
3       For i=0 to len f.mostSplit)
4           base ← f.mostSplit[i]
5           For j=i+1 to len(f.mostSplit)
6               m ← f.mostSplit[j]
7               newFrom ← base+m
8               If newform is either in f.root or f.suffixes
9                   Insert rules for the newForm into rootRules
10                  Find abstract morpheme combination for newForm
11                  Generate filters for newForm
12                  base ← newForm
13              Else
14                  tempForm ← newFrom
15              End If
16              If tempForm != newFrom
17                  Repeat stets 5 to 12 as tempForm is base
18              End If
19          End For
20      End For
```

The root rule generation concludes the preprocessing stage. Operations continue with the parsing. The preprocessing algorithm is given in Algortihm 5.3.

**Algorithm 5.3 Preprocess**

***Input:*** *sentence*
***Output:*** *features, root rules, parsinInput, filters*

```
1  analysedSentence ← conduct morphological analysis on the sentence
2  If all tokens are analysed successfully
3      features ← generate features using analysedSentence
4      parsinInput ← form parsing input using mostSplit for each token
5      filters ← generate filters using features
       rootRules ← generate root rules using features
6  End If
```

### 5.1.2. Parsing

Our parsing algorithm is built on CYK with the extensions of morpheme-level input, root rules and filtering. In this section, we detail the parsing process we have by focusing on the extensions. We continue to use the same example (‘dişi oyan bilir’) in our explanations. Based on the example, step by step, we show the paring table, and we illustrate the construction of parse trees.

Initialization and the main flow of the parsing are the same as traditional CYK. It starts with the initialization of the parsing table and the constituents propagate in the first row. For this example, a 7x7 table is initialized. Next, constituents are retrieved from the unary rules in CNF grammar based on the input. The filters are employed at this moment. During the retrieval of the constituents for the first row, only the constituents matching with the filters are inserted into the table. Additionally, unary root rules are involved in this stage and the special symbols are inserted into the first row of the table. The state of the parsing table after the first row is filled is illustrated in Figure 5.1. At this step of the algorithm, partially constructed parse trees are illustrated in Figure 5.2. Normally, special symbols of the root rules are not included in the parse trees but in the illustrations of this example, we show them on purpose.

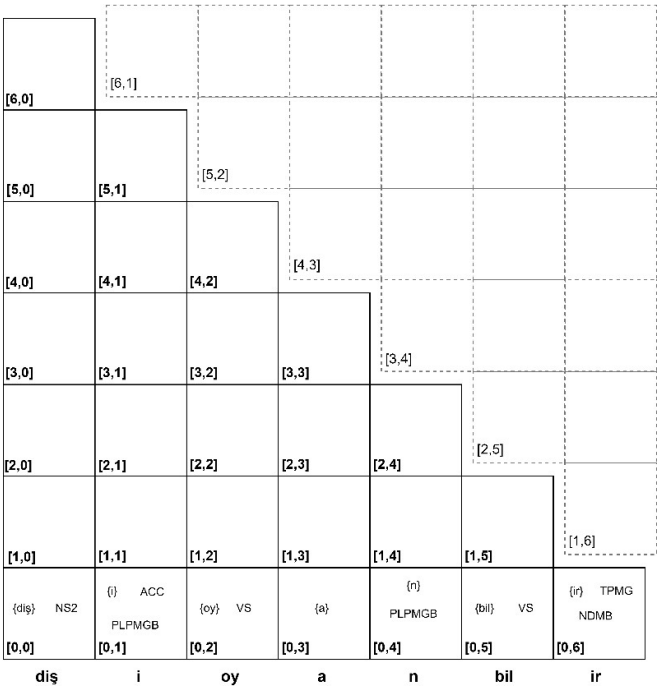


**Figure 5.1** Paring table after the first row is processed.


**Figure 5.2** Partially constructed parse trees after the first row of the parsing table is processed.

Next, a double-nested loop works on the rest of the table. As in the traditional CYK, three cells are selected systematically. Two of them are the sources which provide the constituents to combine and form a possible RHS for grammar rules. The third cell is the target which will get the LHS of all rules that the possible RHS matches. The table is filled with the appropriate constituents based on the grammar rules, in this manner.

We will trace out algorithms as the table is processed line by line. We will only show the constituents which are related to the parses in (5.1).

Next, the second row of the table is filled. In this example, there is one CNF grammar rule on this level. NS2 in the cell [0,0] and ACC in the cell [0,1] are combined into NPACC in [1,0]. The corresponding rule is given in (5.8).

(5.8) NPACC → NS2 ACC

The rest of the rules function here are the root rules. The design we have on the roots rules is totally compatible with this flow. Thus, we can generate various surface forms of each token as the parsing continues. The special symbols we used for the root rules reside with the actual constituents in the cells. In this way, we include such symbols in the process. Yet, we allow special symbols only to be combined with each other to prevent illegal combinations. Whenever the combination contains special symbols, we look for the possible RHS match in the root rules. The LHS of the matching root rule, which is a special symbol, goes through an additional process. First, the special symbol is inserted into the target cell. Thus, it can be used in other root rules further, if necessary. Additionally, the new surface form that the special symbol yields is involved in the parsing table at this moment. Surface forms are terminal symbols in the grammar. They are not located in the parsing table. Therefore, any new surface form we obtain in the middle of the process is not inserted into the table. Like at the initial step of the parsing where we fill the first row of the table, constituents regarding the form are retrieved, filtered and inserted into the target cell.

In the second row, '{diş}' in the cell [0,0] and '{i}' in the cell [0,1] are combined into '{dişi}' and placed into [1,0]. Also, since the new symbol has matching RHS in the CNF grammar, QP is added into the same cell. In the same manner, '{oya}', '{an}' and '{bilir}' are formed. The relevant constituents NS3, and NDMB of the new forms are added into the corresponding cell. Filters of 'an' take action at this moment and allow only NDMB added to the table for 'an'. Normally, '{bilir}' (a nominal stem) has relevant constituents as well but since those constituents are not used in the valid parses of the example, we omit them here. All rules mentioned here are given in (5.9).

(5.9) a. {dişi}→{diş}{i}

QP→dişi

b. {oya}→{oy}{a}

NS3→oya

c. {an}→{a}{n}

NDMB→an

d. {bilir}→{bil}{ir}

The state of the parsing table after the second row is filled is illustrated in Figure 5.3. Partially constructed parse trees are illustrated in Figure 5.4.

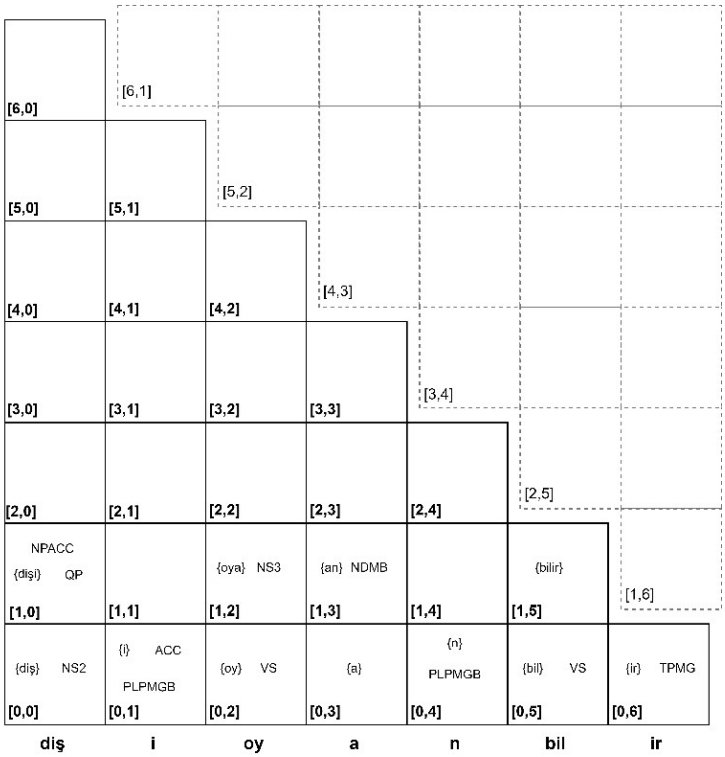


**Figure 5.3** State of the parsing table after the second row is filled.

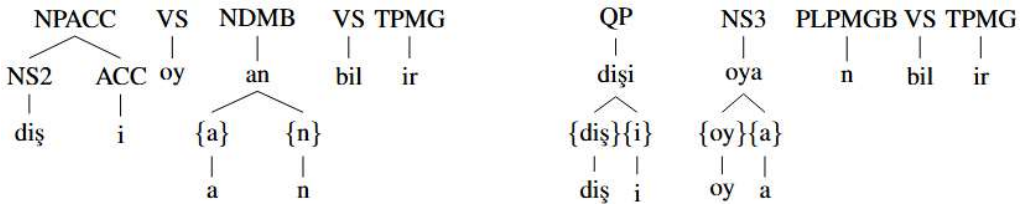


**Figure 5.4** Partially constructed parse trees after the second row of the parsing table is processed.

During the third row is processed, NPACC in the cell [1,0] and VS in cell [0,2] are combined into VS and placed in cell [2,0]. The CNF rule functioning here is given in (5.10). The updated parsing table is illustrated in Figure 5.5.

(5.10) VPS→NPACC VS

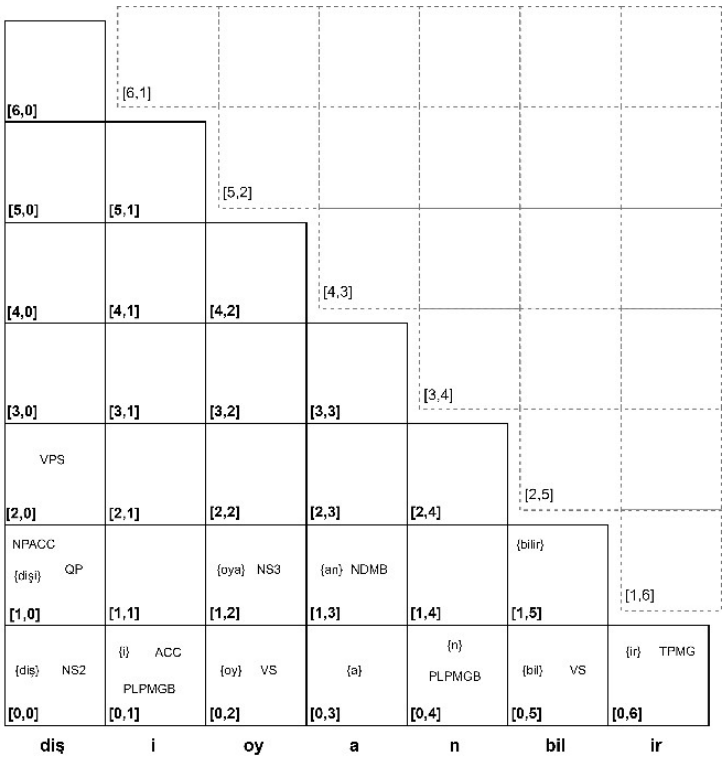


**Figure 5.5** State of the parsing table after the third row is filled.

On the fourth row, NPS3 is formed by QP in the cell [1,0] and NS3 in cell [1,2] and placed into [3,0]. The CNF rule functioning here is given in (5.11). The updated parsing table is illustrated in Figure 5.6. Partially constructed parse trees are illustrated in Figure 5.7.

(5.11) NPS3→QP NS3

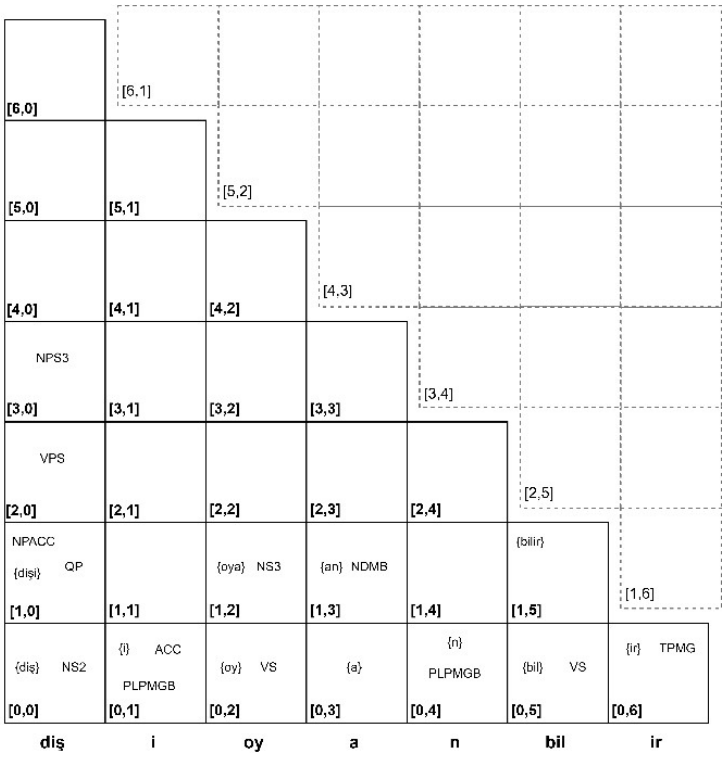


**Figure 5.6** State of the parsing table after the fourth row is filled.

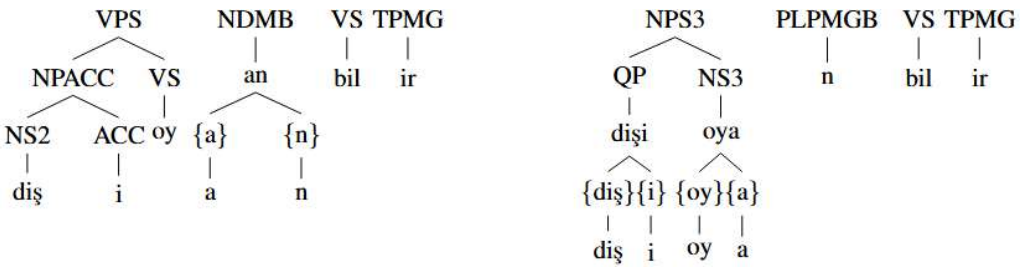


**Figure 5.7** Partially constructed parse trees after the third and fourth rows of the parsing table are processed.

On the fifth row, two different NPSUB constituents are formed. One of them is formed by the combination of NPS3 in the cell [3,0] and PLPMGB in [0,4]. The other one is formed by VPS in [2,0] and NDMB in [1,3]. The CNF rules functioning here are given in (5.12). The updated parsing table is illustrated in Figure 5.8. Partially constructed parse trees are illustrated in Figure 5.9.

(5.12) a. NPSUB→NPS3 PLPMGB

b. NPSUB→VPS NDMB

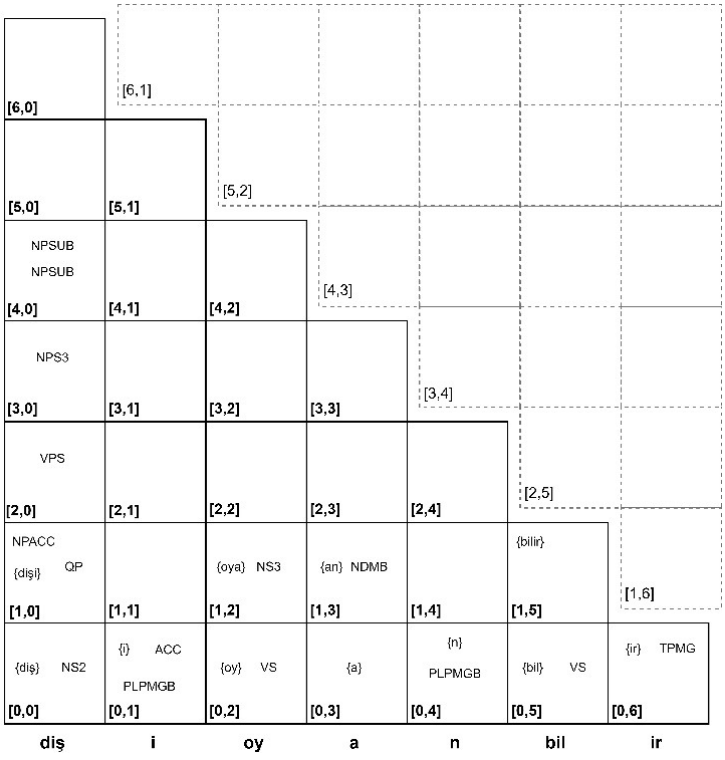


**Figure 5.8** State of the parsing table after the fifth row is filled.

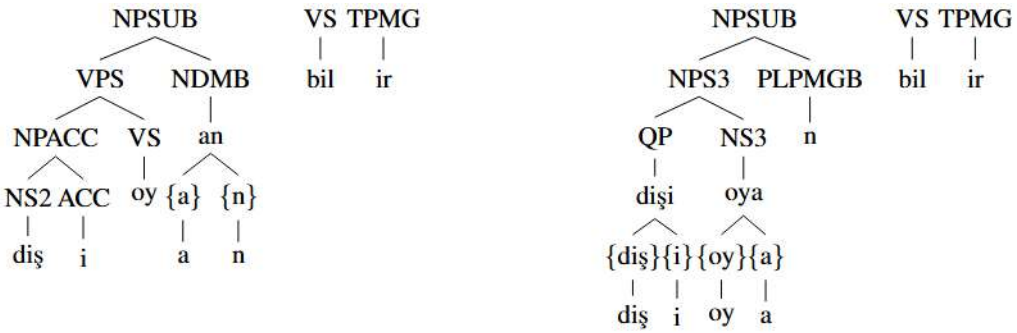


**Figure 5.9** Partially constructed parse trees after the fifth row of the parsing table is processed.

On the sixth row, both NPSUBs0 are combined with VS in the cell [0,5] separately and they form two VPSSUB in cell [5,0]. The CNF rule functioning here is given in (5.13). The updated parsing table is illustrated in Figure 5.10. Partially constructed parse trees are illustrated in Figure 5.11.

(5.13) VPSSUB→NPSUB VS

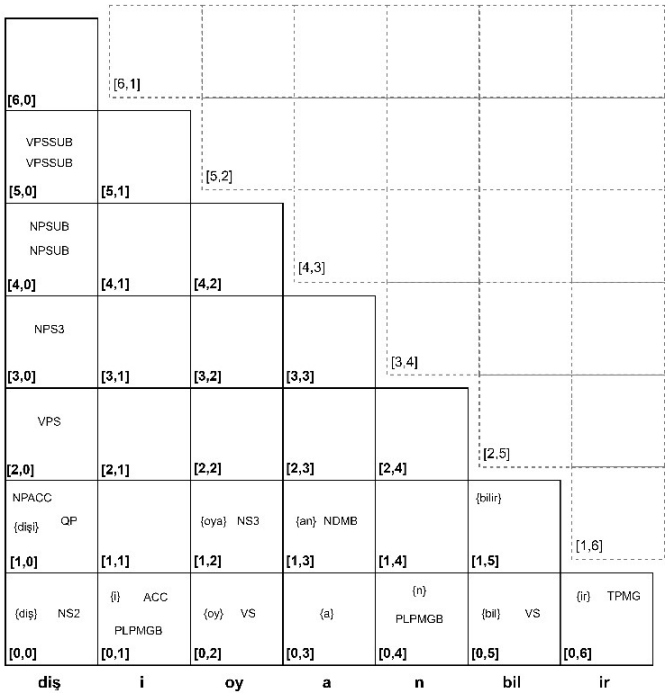


**Figure 5.10** State of the parsing table after the sixth row is filled.

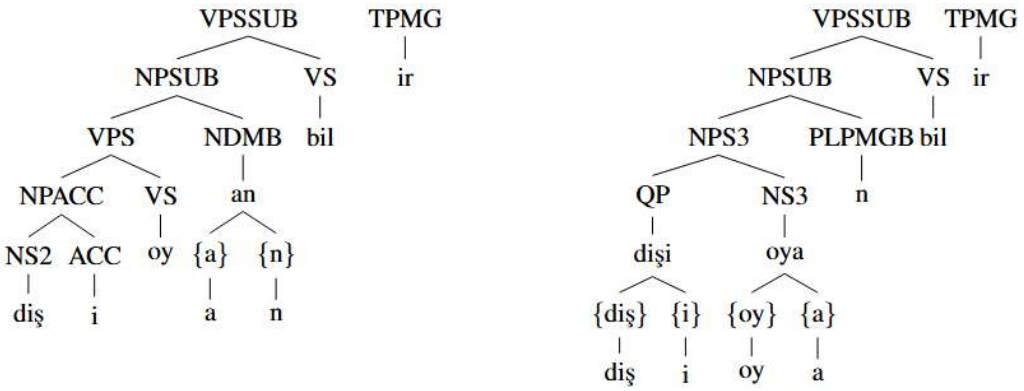


**Figure 5.11** Partially constructed parse trees after the sixth row of the parsing table is processed.

The overall parsing process ends when the top left cell is processed. The existence of the constituent S in that cell indicates if there is a valid parse or not. For each parse, there is a separate S constituent which is the roots. In this example, two S constituents are formed in [6,0] by both VPSSUB in [5,0] and TPMG in [0,6]. The final state of the parsing table is in Figure 5.12. The resulting parse trees are illustrated in Figure 5.13.

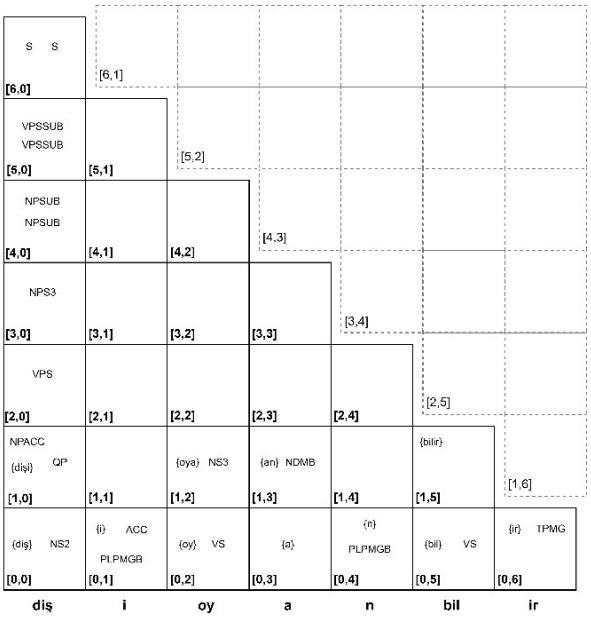


**Figure 5.12** The final state of the parsing table.

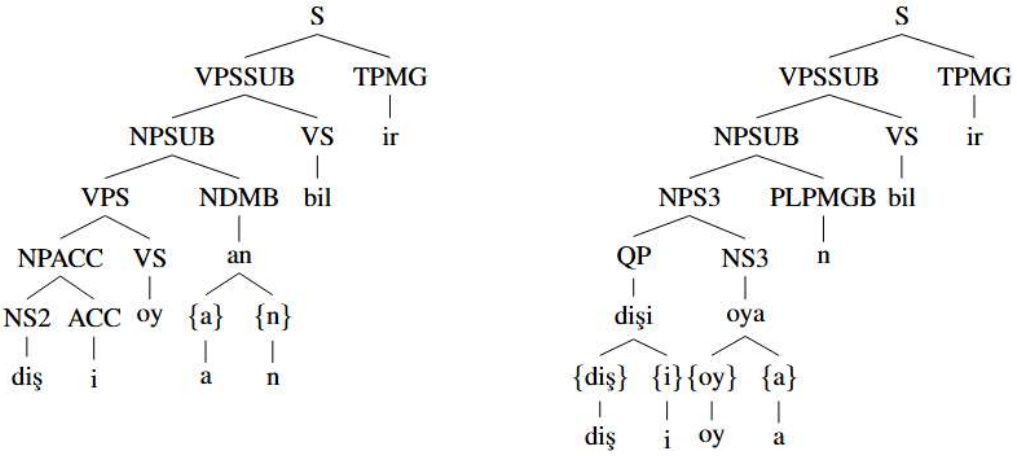


**Figure 5.13** Fully constructed parse trees.

Although the parsing is completed there is one last step. The parse trees we illustrate show the constructed parse tree with respect to the CNF rules. Because, in fact, they are built based on the CNF of the grammar. The last step of the parsing process is to convert the parse trees back to the CFG level. Converted parse trees are illustrated in Figure 5.14 and Figure 5.15. These are the final product of the system. The parsing algorithm is given in Algorithm 5.4.

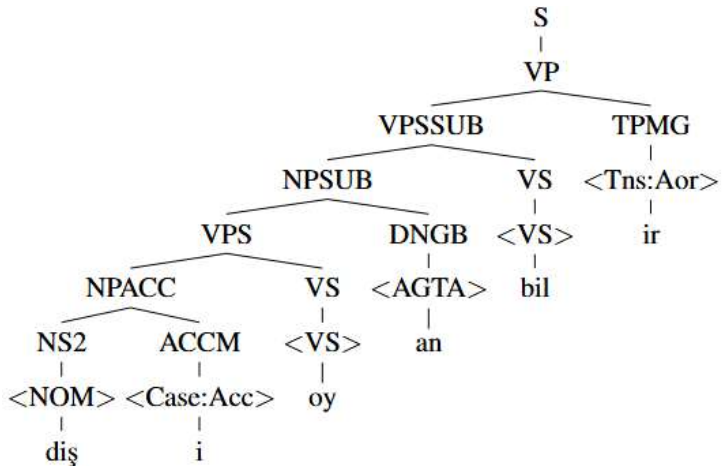


**Figure 5.14** Parse tree of the sentence in (5.1-a).

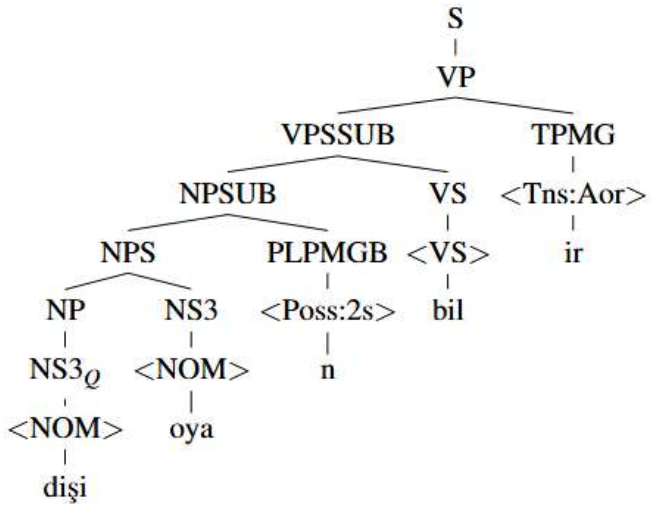


**Figure 5.15** Parse tree of the sentence in (5.1-b).

**Algorithm 5.4 Parsing**

***Input:*** *parsingInput, grammar, rootRules, filters*
***Output:*** *parses*

```
1   Let table be a 2D list
2   For i=0 to len(parsingInput)
3       rhs ← parsingInput[i]
4       get all lhs from grammar[rhs] and rootRules[rhs]
5       filter and insert lhs into table[0][i]
6   End For
7   For i=2 to len(parsingInput)+1
8       For j=1 to len(parsinginput)-i+2
9           For k=1 to i
10              const1 ←get all elements in table[k-1][j-1]
11              const2 ← get all elements in table[i-k-1][j+k-1]
12              For each c1 and c2 in const1×const2
13                  rhs ← c1 + c2
14                  get all lhs from grammar[rhs] and rootRules[rhs]
15                  Insert all lhs into table[i-1][j-1]
16                  If any lhs is special symbol
17                      rhs2 ← convert lhs into terminal symbol
19                      repeat steps 14 and 15 as rhs2 is rhs.
20                  End If
21              End For
22          End For
23      End For
24  End For
```

## 5.2 PERFORMANCE, TESTING AND DISCUSSION

Extending the constituency parsing to the morpheme level increases the computational burden. As an addition to the traditional CYK algorithm, our approach has differences such as preprocessing, operating root rules during parsing and the possible increase in the input length. To explain how the computational complexity changes we did a brief analysis which covers all steps of our approach.

The preprocessing stage consists of major operations such as morphological analysis, most split calculation, listing all roots and affixes, filter generation and root rule generation. The complexity brought by the morphological analysis step is negligibly low since we use an HFST-based analyser. Execution times for HSFT-based analysers are reported in (Silfverberg & Lindén, 2009).

On the other hand, the rest of the preprocessing steps depend on several statistics related to tokens. For a token, the number of surface forms in each analysis affects the complexity of the most split calculation. In the worst case, each analysis of the token has the longest number of surface morphemes possible. Let's call this value $L$ for a given lexicon. Assuming there are $N$ tokens, The complexity becomes $O(NL^2)$. Since $L$ would be a constant value, so the complexity of most split extraction is $O(N)$.

Listing all roots and suffixes depends on the number of morphemes as well. For a given lexicon, let's assume the number of roots for a token is $R$ and again maximum number of surface morphemes is $L$. For $N$ tokens, the time complexities for listing roots and morphemes can be expressed as $O(NL)$ and $O(NR)$ respectively. When the constants are eliminated, complexities for both becomes $O(N)$ and in total, still $O(N)$.

Filter generation focuses on each surface element in each analysis of token. $L$ would be the number of morphemes again and $A$ would be the maximum number of analyses a token from a lexicon can have. The time complexity is $O(NAL)$, $A$ and $L$ would be constants and the complexity is $O(N)$.

The complexity of the root rule generation is based on the number of tokens ($N$) and the length of most split ($S_i$ where i indicates token). Thus, for one token, we can express the complexity of root rule generation as $O(S_i^2)$. Assuming there are $N$ tokens and the upper limit of the most split in the lexicon is M, the complexity becomes $O(NM^2)$. For a given lexicon, the M value would be a constant, therefore, it can be neglected in the complexity analysis. Thus, complexity of the root rule generation becomes $O(N)$.

The total complexity of the preprocessing phase can be expressed as $O(N) + O(2N) + O(N) + O(N) = O(4N)$. This is equivalent to $O(N)$ where $N$ is the number of tokens.

The traditional CYK algorithm is classified as $O(N^3)$ (Kasami, 1966). Our extended CYK algorithm changes the length of the parsing input from $N$ to $M$ where $M = S_1 + S_2 + \cdots + S_N$. Thus, we can classify our parsing phase as $O(M^3)$. If $S$ is the upper bound value for $S_i$, we can have the relation $NS \geq M$. And the time complexity of the overall process would be $O(N) + O(M^3) = O(M^3)$.

We conducted a test to both evaluate the performance and measure the execution time of our parsing system. We run the algorithms on a computer with 2.6 GHz Intel Core i7-10750H CPU and 16 GB RAM. To form a test set, we randomly picked 500 sentences from the same dataset. Explicitly, we did the random selection by excluding the ones we picked for the treebank. The test set we formed has an average number of 8 morphemes in the input. Our system parsed 317 of 500 sentences successfully.

Among not analysed sentences, 26 sentences are not analysed due to the failure of morphologic analysis step. This means we are unable to have morphological analyses for at least one of the words in those sentences and this terminates the parsing process. It requires an update in the morphological analyser to solve this issue. The rest of the fails are due to the absence of required grammar rules. When any grammar rule which is required to construct a specific sentence is not in the grammar, the CYK algorithm fails to yield valid parses for that sentence. The solution for this issue is the addition of the required

rules to the grammar. As the main objective of the future work for the project, we plan to grow the grammar and the treebank with a variety of sentences. In this way we plan to have a more comprehensive grammar and a system which can yield parses for more sentences. An example of sentence which our algorithm fails to yield parses is given in (5.14).

(5.14) yürü-r-ken şarkı mırılda-n-ıyor-du-m
walk-<AgtA>-<IsWhen> song hum-<Pasv>-<Tns:Pres>-<Cpl:Past>-<Prsn:1s>
'I was humming a song while I was walking'

Since the grammar we use does not have all the rules to recognize and parse this sentence, the algorithm can only build the parse tree of the sentence partially. Normally, such a structure is never constructed since it is not complete, but for this example, to demonstrate the issue with this sentence and the solution, we illustrate the partial tree in Figure 5.16.

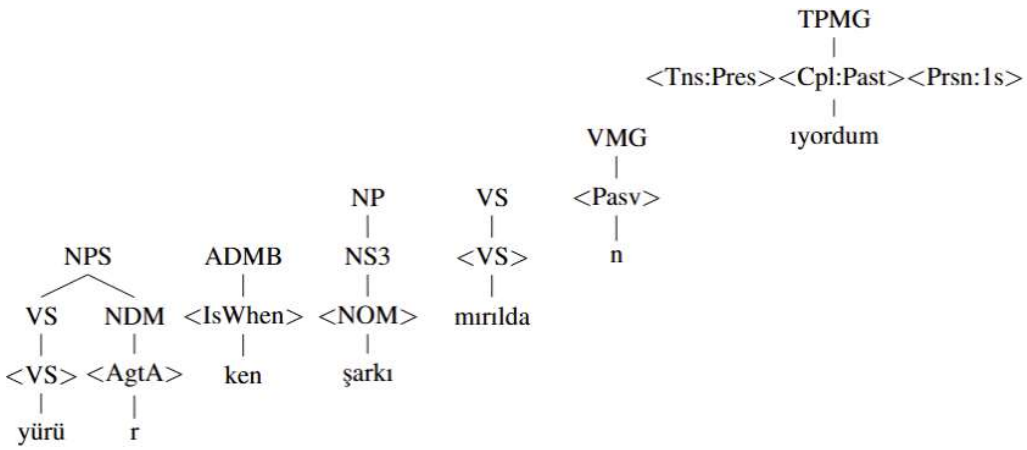


**Figure 5.16** Partial parse tree of the sentence in (5.14).

During the parsing phase, our algorithm cannot construct the full parse tree due to the missing rules in the grammar. Such as the rule which combines a nominal phrase stem and an adverb deriving morpheme to form an adverb phrase. Of course, for the tree illustrated in X, this rule is not enough to

complete the parse tree. In addition, a rule to combine the adverb phrase, the nominal phrase and the verb stem into the verb phrase stem, a rule to combine the verb phrase stem and verbal morpheme into the verb phrase stem and lastly a rule to combine highest verb phrase stem with tense person morphemes into the verb phrase. Such rules are given in (5.15).

(5.15) a. ADVP→NPS ADMB

b. VPS→ADVP NP VS

VPS→VPS VMG

VP→VPS TPMG

Among all four rules given in (5.15), only the one in (5.15-a) is not present in the grammar. The rest which are given in (5.15-b) are already in the grammar. Therefore, with the addition of the rule in (5.15-a) into the grammar, our parser can parse the sentence in (5.14) successfully. The complete parse tree of the sentence is illustrated in Figure 5.17.

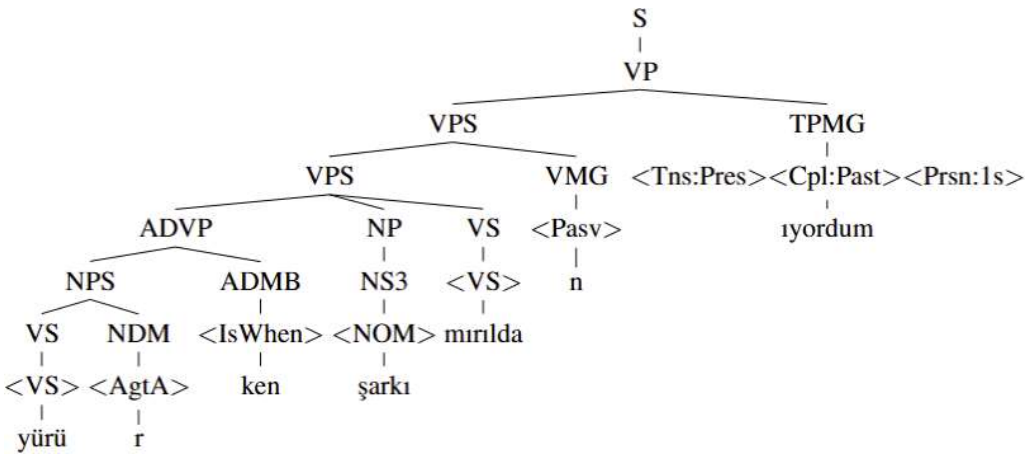


**Figure 5.17** Parse tree of the sentence in (5.14).

During this test, we measured the average CPU time of the preprocessing phase as 1 millisecond and of the parsing phase as 6 milliseconds.

As far as we know, there is no other constituency parser for Turkish. Therefore, we cannot use any other system to compare with ours. Nonetheless,

we can do a comparison on the morpheme level to provide a benchmark. Our system can generate POS tags for each morpheme as a byproduct. Thus, we can use these POS tags to have a comparison with another POS tagger. Zemberek (Akın & Akın, 2007) is a morphological analysis framework which can yield POS tags as well. We used the same test set of 500 sentences and got POS tags from both Zemberek and our system. We checked if the POS tags generated by Zemberek include the ones our system generates for each valid parse. Comparison showed that 85% of the POS tags match Zemberek's.

# CONCLUSION AND SUGGESTIONS

Parsing is an important step in NLP-related tasks. It is a wide study area which involves various methodologies such as constituency and dependency. Among all, constituency parsing is the method which structures the sentence as a hierarchy of constituents. Constituency parsing utilizes context-free grammar which is an implementation of constituency grammar. This grammar formalism expresses syntactic elements such as words, phrases, etc. as constituents and defines how the constituents form the hierarchical structure.

Constituency parsing is under the influence of lightly inflected languages. It has challenges when dealing with a MRL. Therefore, it requires extra care when to adapt the traditional method for such a language.

Turkish is a morphologically rich and agglutinative language which has a relatively free word order. Morphological cues of Turkish have specific roles in the syntax and even significant places in syntactic structure. Since the traditional constituency approach reaches the words at the bottom level, sub-word structures are out of its range. This requires extra steps to involve morphemes in parsing. Also, the free word order of Turkish is another issue for the order-sensitive mechanism of the constituency approach.

In this study, we focus on constituency parsing of Turkish and tackle the challenges of such a language for this task. Out ideas are designed specifically for Turkish, but they can be adapted to any similar language.

NLP studies practice the morphology syntax intersection as a series of various labels in general. Yet also, there is a concrete idea of structuring syntax in terms of morphological elements. Our study focuses on utilizing morphemes in the syntactic structure and even we structure the syntax through morphemes. We do this by involving morphemes in the parsing process.

We enable morpheme-based parsing by extending the process to the morphemes. We implement the extension on the CYK algorithm which is a bottom-up parsing approach. At the bottom, while the traditional CYK starts

parsing from words, our extension replaces the input with a list of morphemes.

The transition from a list of words to a list of morphemes is not a trivial process and has its challenges. It requires conducting a morphological analysis on each token. Since a word may have different analyses, the analysis result must be processed before it is used.

One challenge here is the varying number and positions of morphemes. We handle this issue by unifying the analyses based on the surface forms as much as possible. We make it by attaching zero morphemes to non-zero ones based on pre-determined rules. Yet, this only helps if analyses have matching surface forms.

This brings up another challenge. If the token has analyses with different surface forms, unification does not help. Of course, having a morphological disambiguation is a solution here. However, since the CYK algorithm can yield all possible parses for the given input, we want to keep the morphological ambiguity and transfer it to the syntax level. In such a case, we employ temporary rules to involve each different surface form in the parsing process. These input-specific rules function as morphotactic rules and form up the new surface forms during the natural flow of the CYK algorithm. To enable this, the morpheme-level input must be in a form which allows the construction of different morphemes. We obtained such a form by comparing the surface forms and calculated the most fractured form of the token.

The traditional CYK algorithm operates on the words. Thus, naturally, word boundary information is included in the input. Morpheme-level parsing eliminates the word boundary. This is an issue when a morpheme's homonym is a word. To keep the distinction, we employ filters generated based on the analyses of tokens.

In addition to the parsing extension, we designed a constituency set and a grammar which is aligned with the requirements of the extension. Our set includes stems, affixes and groups of affixes instead of words. We designed the grammar to reflect the morphosyntactic structure of Turkish and to utilise the constituents most efficiently. We detailed the grammar and categorized the

constituents to express structures such as the phrases headed by stems, affix attachments to both simple and complex phrases, agreement, different functions of noun phrases based on the case mark, the word order, etc.

To demonstrate the application of our parsing and grammar, we built a small treebank of manually annotated sentences. We designed and implemented a custom tool to facilitate the annotation process. The tool employs morphological analysis, visual support and handy functionalities for parse tree construction. Additionally, it can yield a context-free grammar based on the annotated trees.

# REFERENCES


Abbas, Q. (2016). Morphologically rich Urdu grammar parsing using Earley algorithm. *Natural Language Engineering*, *22*(5), 775.

Akın, A. A., & Akın, M. D. (2007). Zemberek, an open source NLP framework for Turkic languages. *Structure*, *10*(2007), 1–5.

Babarczy, A., Gábor, B., Hamp, G., Rung, A., & Szakadát, I. (2005). Hunpars: a rule-based sentence parser for Hungarian. *Proceedings of the 6th International Symposium on Computational Intelligence*.

Bohnet, B., Nivre, J., Boguslavsky, I., Farkas, R., Ginter, F., & Hajič, J. (2013). Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, *1*, 415–428.

Bozsahin, C. (2002). The combinatory morphemic lexicon. *Computational Linguistics*, *28*(2), 145–186.

Brants, S., Dipper, S., Hansen, S., Lezius, W., & Smith, G. (2002). The TIGER Treebank. *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, 24–41.

Çakıcı, R. (2009). *Wide-coverage parsing for Turkish*. Unpublished PhD Thesis, University of Edinburgh.

Can, B., Aleçakır, H., Manandhar, S., & Bozşahin, C. (2022). Joint learning of morphology and syntax with cross-level contextual information flow. *Natural Language Engineering*, *28*(6), 763–795.

Çetinoğlu, Ö., & Kuhn, J. (2013). Towards joint morphological analysis and dependency parsing of Turkish. *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, 23–32.

Çetinoğlu, Ö., & Oflazer, K. (2018). Deep Parsing of Turkish with Lexical-Functional Grammar. In K. Oflazer & M. Saraçlar (Eds.), *Turkish Natural Language Processing* (175–206). Springer International Publishing.

Charniak, E. (2000). A maximum-entropy-inspired parser. *In Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, 132–139.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, *2*(3), 113–124.

Chomsky, N. (1965). *Aspects of the Theory of Syntax* (Issue 11). MIT press.

Chomsky, N., & Miller, G. A. (1963). Introduction to the formal analysis of natural languages. *Journal of Symbolic Logic*, *33*(2), 269–322.

Cocke, J., & Schwartz, J. T. (1970). *Programming languages and their compilers.* Courant Inst. Math. New York University.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. *In Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 16–23.

Çöltekin, Ç. (2015). A grammar-book treebank of Turkish. *In Proceedings of the 14th Workshop on Treebanks and Linguistic Theories (TLT 14)*, 35–49.

Dayanik, E., Akyürek, E., & Yuret, D. (2018). MorphNet: A sequence-to-sequence model that combines morphological analysis and disambiguation. *CoRR*, *abs/1805.07946*.

De Marneffe, M.-C., & Nivre, J. (2019). Dependency grammar. *Annual Review of Linguistics*, *5*(1), 197–218.

Dönmez, İ., & Adalı, E. (2018). Contex Free Grammer For Turkish. *Süleyman Demirel University Journal of Natural & Applied Sciences*, *22*(2), 552–561.

Earley, J. (1970). An efficient context-free parsing algorithm. *Commun. ACM*, *13*(2), 94–102. https://doi.org/10.1145/362007.362035

Erguvanli, E. E. (1984). *The Function of Word Order in Turkish Grammar* (Issues 106-107. c.ler). University of California Press.

Erkman-Akerson, F. (1994). Ad tümcesindeki yalın tümlecin yönetici ad olma koşulları. *Dilbilim Araştırmaları Dergisi*, *5*, 62–79.

Eryigit, G. (2012). The Impact of Automatic Morphological Analysis & Disambiguation on Dependency Parsing of Turkish. *LREC*, 1960–1965.

Eryiğit, G., Nivre, J., & Oflazer, K. (2008). Dependency parsing of Turkish. *Computational Linguistics*, *34*(3), 357–389.

Fraser, A., Schmid, H., Farkas, R., Wang, R., & Schütze, H. (2013). Knowledge sources for constituent parsing of German, a morphologically rich and less-configurational language. *Computational Linguistics*, *39*(1), 57–85.

Göksel, A., & Kerslake, C. (2004). *Turkish: A comprehensive grammar*. Routledge.

Goldberg, Y. (2011). *Automatic syntactic processing of Modern Hebrew*. Ben Gurion University of the Negev.

Hankamer, J. (2004). Why there are two -ki's in Turkish. In K. İmer & G. Doğan (Eds.), *Current research in Turkish linguistics*. Eastern Mediterranean University.

Johansson, R., & Nugues, P. (2007). Extended constituent-to-dependency conversion for English. *In Proceedings of the 16th Nordic Conference of Computational Linguistics*, (105–112). University of Tartu.

Jurafsky, D., & Martin, J. H. (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/

Karabulut, F. (2009). Ad Öbeği Taşınımı ve Boşluk Kuramı Bağlamında. *Uluslararası Türk Dili ve Edebiyatı Kongresi (UTEK)*, (261–299). İstanbul Kültür Üniversitesi.

Kasami, T. (1966). An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report No. R-257*.

Kayadelen, T., Öztürel, A., & Bohnet, B. (2020). A gold standard dependency treebank for Turkish. *In Proceedings of the 12th Language Resources and Evaluation Conference*, 5156–5163.

Kim, M., & Park, J. (2022). A note on constituent parsing for Korean. *Natural Language Engineering*, *28*(2), 199–222.

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. *In Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, 423–430.

Koç, N. (1990). *Yeni dilbilgisi*. İnklâp Kitabevi.

Kornflit, J. (1996). On Some Infinitival Wh-constructions in Turkish. *Dilbilim Araştırmaları Dergisi*, *7*, 192–215.

Malkoç, M. (2011). Direkt Bileşenler Analizi. *Journal of Turkish Studies*, *6*(2), 645–656. https://doi.org/http://dx.doi.org/10.7827/TurkishStudies.2204

Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: the penn treebank. *Comput. Linguist.*, *19*(2), 313–330.

More, A., Seker, A., Basmova, V., & Tsarfaty, R. (2019). Joint transition-based

models for morpho-syntactic parsing: Parsing strategies for MRLs and a case study from modern Hebrew. *Transactions of the Association for Computational Linguistics*, *7*, 33–48. https://doi.org/10.1162/tacl_a_00253

Özenç, B., & Solak, E. (2019). Syntactic annotation of Turkic languages. *In Proceedings of the International Conference on Turkic Languages Processing (TURKLANG)–Simferopol*.

Özenç, B., & Solak, E. (2020). Visual modeling of Turkish morphology. *In Proceedings of the 12th Language Resources and Evaluation Conference*, 3984–3990.

Silfverberg, M., & Lindén, K. (2009). Hfst runtime format—a compacted transducer format allowing for fast lookup. *Finite-State Methods and Natural Language Processing*, *13*.

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013). Parsing with Compositional Vector Grammars. In H. Schuetze, P. Fung, & M. Poesio (Eds.), *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (455–465). Association for Computational Linguistics. https://aclanthology.org/P13-1045

Tesnière, L. (2015). *Elements of structural syntax*. John Benjamins Publishing Company.

Tuç, S. (2020). *Neural dependency parsing for Turkish*. Unpublished Master's thesis, Haccettepe University, Graduate School of Science and Engineering.

Türk, U., Atmaca, F., Özateş, Ş. B., Berk, G., Bedir, S. T., Köksal, A., Başaran, B. Ö., Güngör, T., & Özgür, A. (2022). Resources for Turkish dependency parsing: Introducing the BOUN treebank and the BoAT annotation tool. *Language Resources and Evaluation*, *56*(1), 259–307.

Underhil, R. (1985). *Turkish grammar*. MIT Press.

Uzun, N. E. (2000). *Evrensel Dilbigisi ve Türkçe*. Multilingual.

Yamada, K., & Knight, K. (2001). A syntax-based statistical translation model. *In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 523–530.

Yildiz, O., Solak, E., Çandır, Ş., Ehsani, R., & Görgün, O. (2015). Constructing a Turkish Constituency Parse TreeBank. *Information Sciences and Systems 2015: 30th International Symposium on Computer and Information Sciences*, *363*, 339–347.

Younger, D. H. (1967). Recognition and parsing of context-free languages in time n3. *Information and Control*, *10*(2), 189–208.

# APPENDICES

## APPENDIX A.1

| Morpheme Constituents | | Phrase Constituents | |
|---|---|---|---|
| **Constituent** | **Description** | **Constituent** | **Description** |
| VS | Verbal stem | S | Sentence |
| TPMG | Tense and Person morphemes group | VPS | Verbal phrase stem |
| VMG | Verbal morphemes group | VPSSUB | Verbal phrase stem, with subject |
| NS1,NS2,NS3 | Nominal stem | NPS1,NPS2,NPS3 | Nominal phrase stem |
| PLPMG, PLPMGB | Plural and Possessive morphemes group | NP | Nominal phrase functioning as object |
| CMG, CMGB | Instrumental, Ablative Case morphemes group | NPSUB | Nominal phrase functioning as subject |
| ACC | Accusative Case morpheme | QP | Qualifier phrase |
| DAT | Dative Case morpheme | NPRED | Nominal predicate |
| GEN | Genitive Case morpheme | NPDAT | Dative marked nominal phrase |
| ADV | Adverb | NPGEN | Genitive marked nominal phrase |
| ADJ | Adjective | NPACC | Accusative marked nominal phrase |
| ADM | Adverb deriving morpheme | NPC | Case marked nominal phrase except Dative, Genitive and Accusative cases |
| VDM | Verb deriving morpheme | ADVP | Adverb phrase |
| NDM, NDMB | Nominal deriving morpheme | | |

# CURRICULUM VITAE

## *Publications*

[1] Ertopçu, B., Kanburoğlu, A. B., Topsakal, O., Açıkgöz, O., Gürkan, A. T., Özenç, B., ... & Yıldız, O. T. (2017, October). A new approach for named entity recognition. In 2017 International Conference on Computer Science and Engineering (UBMK) (pp. 474-479). IEEE.

[2] Topsakal, O., Açıkgöz, O., Gürkan, A. T., Kanburoğlu, A. B., Ertopçu, B., Özenç, B., ... & Yıldız, O. T. (2017, October). Shallow parsing in Turkish. In 2017 International Conference on Computer Science and Engineering (UBMK) (pp. 480-485). IEEE.

[3] Açıkgöz, O., Gürkan, A. T., Ertopçu, B., Topsakal, O., Özenç, B., Kanburoğlu, A. B., ... & Yıldız, O. T. (2017, October). All-words word sense disambiguation for Turkish. In 2017 International Conference on Computer Science and Engineering (UBMK) (pp. 490-495). IEEE.

[4] Ehsani, R., Özenç, B., & Solak, E. (2017, September). A fst description of noun and verb morphology of azarbaijani turkish. In *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing (FSMNLP 2017)* (pp. 62-68).

[5] Özenç, B., & Solak, E. (2017, May). Rule based automatic news tagging. In *2017 25th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). IEEE.

[6] Özenç, B., Ehsani, R., & Solak, E. (2018, November). Moraz: an open-source morphological analyzer for Azerbaijani Turkish. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 25-29).

[7] Özenç, B., & Solak, E. (2020, May). Visual modeling of turkish morphology. In *Proceedings of the Twelfth Language Resources and Evaluation Conference* (pp. 3984-3990).

[8] Özenç, B., & Solak, E. A systematic analysis into the morphotactics of Person, Question and multiple Tenses in Turkish. In *International Conference on Turkic Languages Processing (TURKLANG)–Simferopol*.

[9] Özenç, B., & Solak, E. Visual modelling of morphology. In *International Conference on Turkic Languages Processing (TURKLANG)–Simferopol*.

[10] Özenç, B. (2017). *Morphological analyser for Turkish* (Master's thesis, Fen Bilimleri Enstitüsü).

[11] Özenç, B., & Solak, E. (2019). Syntactic annotation of Turkic languages. In *International Conference on Turkic Languages Processing (TURKLANG)–Simferopol*.

[12] Özenç, B., & Solak, E. (t.y.). TÜRKÇE İÇİN BİÇİMBİRİM TEMELLİ BİR BİLEŞEN GRAMERİ YAKLAŞIMI. Beykoz Akademi Dergisi. https://doi.org/10.14514/beykozad.1508348