

IMPROVED TREATMENT OF WORD MEANING
IN A TURKISH CONVERSATIONAL AGENT

by

Şeniz Demir

B.S. in Computer Engineering, Marmara University, 2000

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Boğaziçi University

2003

IMPROVED TREATMENT OF WORD MEANING
IN A TURKISH CONVERSATIONAL AGENT

APPROVED BY:

Prof. Dr. A.C. Cem Say
(Thesis Supervisor)

Assist. Prof. Dr. Tunga Güngör

Prof. Dr. A. Sumru Özsoy

DATE OF APPROVAL: .././...

ACKNOWLEDGMENTS

First of all I would like to thank Prof. Dr. A.C. Cem Say, my advisor, for his support throughout the thesis work, his patience in the writing stage and his great instructive nature from the beginning until the end.

Special thanks are due to Prof. Dr. A. Sumru Özsoy and Assist. Prof. Tunga Güngör, for their comments on the manuscript of the thesis and their kindly attendance to examination jury. I would also like to thank Prof. Dr. A. Sumru Özsoy for her inputs and advices on the linguistic issues of the study.

I would like to express my gratitude to one of my best friends, Şadi Evren Şeker, for his kindness, his patience and the beauty that we share throughout these years.

Finally, I want to thank my parents, my brother and my grandmother. Without their endless support and love for me, I would never achieve my current position.

ABSTRACT

IMPROVED TREATMENT OF WORD MEANING IN A TURKISH CONVERSATIONAL AGENT

Natural Language Processing (NLP) is a research area of artificial intelligence whose aim is to make computers use natural languages like human beings. The work reported in this thesis is the design and implementation of a Turkish conversational agent whose treatment of word meaning is strengthened and improved with respect to other agents.

Our conversational agent is based on the TOY NLP infrastructure developed in Boğaziçi University. The agent can perform two basic operations: acquiring knowledge from a sentence or answering user queries. The processing of a sentence goes through the morphological, syntactic and semantic levels. If a sentence is morphologically and syntactically checked, its internal semantic representation is derived. If the agent acquires knowledge from this representation, it is stored in its knowledge base. The user can query the agent's knowledge base by asking yes/no questions.

The conversational agent has the capability of learning words that it does not recognize in the input sentence. Beside the word's morphological features, the agent learns what this word means in the context of other concepts in its knowledge base from the user and uses this information in the operations.

The conversational agent has some commonsense knowledge about the external world similar to human beings, unlike its predecessors. This knowledge is represented with a semantic network whose design is open to additions and modifications. This network is used for anaphora resolution and improved handling of questions requiring treatment of deontic modalities.

ÖZET

SÖZCÜK ANLAMLARI İÇİN GELİŞTİRİLMİŞ GÖSTERİMLER KULLANAN BİR TÜRKÇE DİYALOG ETMENİ

Yapay zekanın bir araştırma dalı olan Doğal Dil İşleme'nin (DDİ) amacı bilgisayarların insanlar gibi doğal dilleri kullanabilmesini ve öğrenmesini sağlamaktır. Bu tezde anlatılan çalışma, sözcük anlamları için geliştirilmiş gösterimler kullanan bir Türkçe diyalog etmeninin kurulması ve uygulanmasıdır.

Diyalog etmeni Boğaziçi Üniversitesi'nde geliştirilmiş TOY doğal dil işleme altyapısının üzerine inşa edilmiştir. Temelde iki ana işlem yapabilir: Cümleden bilgi edinme ve diyalog işleme. Cümlenin işlenmesi biçimbilimsel, sözdizimsel ve anlambilimsel aşamalardan geçerek gerçekleştirilir. Cümle biçimbilimsel ve sözdizimsel çözümlmeden geçtikten sonra anlamsal gösterimi elde edilir. Bu gösterimden çıkartılan bilgi bilgi tabanında tutulur. Kullanıcı evet/hayır soruları yardımıyla bilgi tabanındaki bu bilgileri sorgulayabilir.

Diyalog etmeni girilen cümlede yer alan ve bilmediği kelime ya da kelimeleri öğrenebilme yeteneğine sahiptir. Etmen kelimenin biçimbilimsel özelliklerinin yanısıra kelimenin ne anlama geldiğini bilgi tabanındaki kavramlar bağlamında kullanıcıdan öğrenir ve bu bilgiyi işlemleri sırasında kullanır.

Diyalog etmeni kendinden önceki etmenlerden farklı olarak insanlarda olduğu gibi dış dünya hakkında genel bir bilgi dağarcığına sahiptir. Bu dağarcık anlambilimsel bir ağ ile etmene eklenmiş olup yeni eklemelere ve değişikliklere açık bir yapıdadır. Bu anlambilimsel ağ gönderim çözümünde ve yükümlülük kipi içeren soruların cevaplanmasında kullanılmıştır.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iii
ABSTRACT.....	iv
ÖZET.....	v
LIST OF FIGURES.....	viii
LIST OF TABLES.....	xii
LIST OF SYMBOLS/ABBREVIATIONS.....	xiii
1. INTRODUCTION.....	1
1.1. Outline of the Thesis.....	2
2. LITERATURE SURVEY.....	4
2.1. Conversational Systems.....	4
2.1.1. The LOLITA Natural Language Processing System.....	4
2.1.2. GALAXY.....	5
2.1.3. TRIPS: An Integrated Intelligent Problem-Solving Assistant.....	6
2.1.4. TOY.....	8
2.2. Commonsense Knowledge Representation and Lexical Reference Systems.....	12
2.2.1. Cyc.....	13
2.2.2. WordNet.....	15
2.2.3. Role Assignments Model.....	17
2.3. Anaphora Resolution Methods.....	19
3. OVERVIEW OF TOYagent.....	23
3.1. Definitions of Levels and Modules.....	24
3.2. First Order Predicate Calculus (FOPC).....	26
3.3. Semantic Formula of a Sentence.....	28
3.4. Semantic Network Hierarchy.....	30
3.5. Processing of Sentences by TOYagent.....	31
4. LEXICON AND MORPHOLOGY.....	36
4.1. Lexicon.....	36
4.2. Morpheme Structure.....	37
4.2.1. Semantic Representations of Syntactic Categories in the Lexicon.....	39
4.2.2. Semantic Classes and Distinctive Features.....	48

4.3. Generation and Parsing	53
5. ACQUIRING KNOWLEDGE FROM THE USER.....	61
5.1. Lexical Preprocessor.....	62
5.1.1. Stem Checking Module	63
5.1.2. Stem Addition Module	64
5.1.3. Obtaining a Description of the Unknown Word	70
5.2. DCG Parsing of a Sentence.....	74
5.2.1. Noun Phrases (NP and NP_1).....	78
5.2.2. Verb Phrases (VP).....	79
5.2.3. The Semantic Formula of a Sentence.....	80
5.3. Modifications before Semantic Transformation.....	84
5.3.1. Removal of the Conjunction “Hariç” (Except).....	84
5.3.2. Negative Knowledge Modification	84
5.3.3. Anaphora Resolution	85
5.4. Incorporation of Knowledge	90
5.4.1. Simple Sentence Transformation	90
5.4.2. Conjoined Sentence Transformation.....	94
5.4.3. Participle Clause Transformation.....	99
5.4.4. Transformation Postprocessing.....	100
6. QUESTION ANSWERING IN TOYagent	101
6.1. Same Semantic Formula Mechanism (SSF).....	102
6.2. Treatment of Modalities in Question Answering	103
6.3. Explanation Process and Sentence Generation	107
7. CONCLUSION	115
7.1. TOYagent.....	115
7.2. Future Work	117
APPENDIX A: AN EXAMPLE DIALOGUE WITH THE AGENT	120
APPENDIX B: SYSTEM INTERFACE.....	124
APPENDIX C: IMPLEMENTATION ENVIRONMENT	126
REFERENCES	127
REFERENCES NOT CITED	130

LIST OF FIGURES

Figure 2.1.	Example piece of semantic network from LOLITA.....	4
Figure 2.2.	Semantic frame example from GALAXY	6
Figure 2.3.	Output of the parser of TRIPS.....	7
Figure 2.4.	Target scenario of TOY	9
Figure 2.5.	A conversation example from Ögün's conversation system.....	10
Figure 2.6.	Results of a query given to TuSA.....	12
Figure 2.7.	Semantic network example	13
Figure 2.8.	Definition of the constant 'skin' in the Cyc ontology	14
Figure 2.9.	Data sample from Euro WordNet.....	16
Figure 3.1.	General schema of TOYagent	23
Figure 3.2.	Example semantic formula.....	29
Figure 3.3.	Semantic network hierarchy.....	30
Figure 3.4.	Dialogue example	32
Figure 3.5.	Dialogue example (continued)	33
Figure 3.6.	Knowledge assertion example.....	34

Figure 3.7. Answer formation example	35
Figure 4.1. Improved semantic network hierarchy	51
Figure 4.2. Finite state machines for nouns	57
Figure 4.3. Finite state machines for verbs	58
Figure 4.4. Finite state machined for verbs (continued).....	59
Figure 4.5. Parsing a word	56
Figure 4.6. Generating a word	56
Figure 4.7. Parsing an ambiguous word	60
Figure 5.1. TOYagent interface	61
Figure 5.2. Sample dialogue	70
Figure 5.3. Sample dialogue (continued).....	71
Figure 5.4. Sample dialogue (continued).....	72
Figure 5.5. Sample dialogue (continued).....	73
Figure 5.6. Turkish phrase structure rules used in the thesis	76
Figure 5.7. DCG rules used in the construction with the semantic formula.....	82
Figure 5.8. Semantic formula of the sentence “Ehliyeti olan bir kişi araba kullanır”	83
Figure 5.9. Replacement of an anaphor in a semantic formula.....	85

Figure 5.10.	Algorithm of the anaphora resolution method	88
Figure 5.11.	Multiple candidate referents example.....	89
Figure 5.12.	Singsome expression example.....	91
Figure 5.13.	Plurall expression example.....	92
Figure 5.14.	Definitely quantified object example.....	93
Figure 5.15.	Nested quantifier example.....	93
Figure 5.16.	've' (and) conjunction example.....	94
Figure 5.17.	'ise' (if) conjunction example	95
Figure 5.18.	'ise' (if) conjunction example-2	96
Figure 5.19.	've' (and) and 'ise' (if) conjunctions example	97
Figure 5.20.	'veya' (or) and 'ise' (if) conjunctions example.....	98
Figure 5.21.	Semantic representation of the participle clause "çocuğu olan"	99
Figure 5.22.	Participle clause transformation	100
Figure 6.1.	Contents of an example general information list.....	102
Figure 6.2.	A sample rule and some predicates in the knowledge base	104
Figure 6.3.	Sample dialogue using the MSF mechanism	106
Figure 6.4.	Sample dialogue using the DOSSF mechanism	107

Figure 6.5.	Sample dialogue where the mode of the agent changes	108
Figure 6.6.	Parsing of the words “yemek” (meal) and “yemekler” (meals)	110
Figure 6.7.	Retrieving of the word “yemek” (foods) from the “parse” predicate	110
Figure 6.8.	Retrieving of the verb “pişirir” (foods) from the “parse” predicate	111
Figure 6.9.	Sentence construction example	112
Figure 6.10.	Content of the knowledge base during a sample execution	113

LIST OF TABLES

Table 2.1.	Feature dimensions and values of nouns from role assignment model.....	18
Table 2.2.	Feature dimensions and values of verbs from role assignment model	18
Table 2.3.	Vector of the word “ball” from role assignment model.....	18
Table 2.4.	Vectors of the verb “broke” from role assignment model.....	19
Table 2.5.	Important factors in anaphora resolution.....	20
Table 3.1.	FOPC constituents and their uses in language processing	26
Table 4.1.	Semantic representations of question words	40
Table 4.2.	Semantic class abbreviations	46
Table 4.3.	Distinguishable features and abbreviations	47
Table 4.4.	Definitions of nodes and arcs in FSMs	54
Table 5.1.	Stemming operation on the word “leoparı” (the leopard+ACC).....	65
Table 5.2.	Noun phrase structures	78
Table 5.3.	Verb phrase structures.....	79
Table 6.1.	Tense and deontic modality table	105
Table 6.2.	Abbreviations of the answers.....	105

LIST OF SYMBOLS/ABBREVIATIONS

\wedge	Conjunction
\rightarrow	Implication
\forall	Universal Quantifier
\exists	Existential Quantifier
λ	Notation Used for Lambda Calculus
ADJ	Adjective
ADJ_1	Adjective(verb)
AI	Artificial Intelligence
CNP	Conjoined Noun Phrase
CONJ	Conjunction
D	Determiner
DCG	Definite Clause Grammars
DOSSF	Deduction on Similar Semantic Formula
FOPC	First-order Predicate Calculus
FSM	Finite State Machine
IDPR	Indefinite Pronoun
KB	Knowledge Base
MSF	Modified Semantic Formula
N	Noun
N1	Noun(nominal verb)
NLP	Natural Language Processing
NP	Noun Phrase
NP_1	Noun Phrase(with nominal verb)
NX	Noun Extension
PN	Proper Noun
QP	Question Phrase
S	Sentence
S1	Sentence(s)

SC	Conjoined Sentence
SSF	Same Semantic Formula
TOY	Türkçe Okur Yazar
TOY	Türkçe Okur Yazar agent
TU	Time Unit
V	Verb
VP	Verb Phrase

1. INTRODUCTION

Natural Language Processing (NLP) is a developing research area of artificial intelligence (AI) which deals with communication of computer systems and human beings via natural language sentences. Systems with flexible natural language interfaces would have clear user-friendliness advantages over others. To produce human-like dialogue, a program would have to understand natural language sentences. Understanding means recognizing the information expressed by these sentences.

There are five basic levels at which natural language can be processed:

The phonetic level deals with the usage of the sounds in a language. The sound alphabet and their physical realizations and the interpretation of sounds across the words are the subject of this level.

The morphological level deals with the construction of the words from roots and suffixes (morphemes). A morpheme is the primitive unit of meaning in a language (roots, affixes). Each word is constructed by the affixation of suffixes to the root. The order of these suffixes, their forms (the inflectional suffixes, derivational suffixes, etc.) and the rules governing the word construction are also the subject of this level.

The syntactic level deals with the construction of correct sentences from the words and their structural roles in these sentences. The structure rules of the language are also the subject of this level.

The semantic level deals with the meanings of words and the meanings of sentences. The meaning of a sentence is independent of the context in which it is used and formed by combining the meanings of its constituents (words). Ambiguity in word meaning is also analyzed in this level. Meanings are defined by one or more internal semantic representations associated with each word in the lexicon. The meaning extraction process is also a subject of this level.

The pragmatic level deals with the purposeful usage of language in context. It is concerned with the effects of the external world in understanding the meaning of sentences.

A conversational agent is a kind of software agent and has the capability of communicating with users through its user-friendly interface. Conversational agents are designed to perform tasks given as user directives. The tasks that conversational agents can perform and the mechanisms they possess, like question answering, information extraction, deduction, etc. vary according to the areas in which they are used.

The aim of the study reported in this thesis is to develop a Turkish conversational agent whose treatment of word meaning is improved in several respects with respect to its predecessors. The agent manages dialogues with the user and decides what to do according to the user inputs. It either acquires the knowledge contained in the input sentences or answers the user's questions.

If this agent does not recognize a word in an input sentence, it redirects the dialogue to try to learn the structural and semantic features of that word. The agent has a rudimentary store of commonsense knowledge about the external world. Using this knowledge, the agent can make improved deductions on the knowledge it has. Other new features of the agent include a treatment of questions involving different types of deontic modalities.

Our work uses the natural language processing infrastructure developed by Çetinoğlu [1] as its basis. We embedded a commonsense knowledge base to this infrastructure. The main application developed by Çetinoğlu, which is called TOY, enables the user to ask questions and to give new information to the program at runtime. The conversational agent that we developed, which we named TOYagent, is based on this program.

1.1. Outline of the Thesis

Chapter 2 gives brief summaries of related work on conversational systems, commonsense knowledge representation, lexical reference systems and anaphora

resolution methods. Chapter 3 gives an overview of the system architecture and meaning representation used by the program. Chapter 4 explains the morphological level of the agent. The morpheme representation used in the lexicon and the method used for generating or parsing the words are presented. In Chapter 5, the syntactic and semantic levels of the agent, as well as the process of acquiring knowledge contained in a sentence are explained. Chapter 6 explains the ways of querying the agent's knowledge. The answer formation and sentence generation capabilities of the agent are explained. The last chapter summarizes the work done and points to possible future developments.

Appendix A includes an example dialogue with the agent. The system interface is explained in Appendix B. Appendix C gives information about the implementation environment.

In the example dialogues, the sentences entered by the user are given in bold throughout the thesis.

2. LITERATURE SURVEY

In this chapter, some concepts and previous work related to our thesis will be discussed.

2.1. Conversational Systems

Before going on to the details of our conversational agent, some conversational systems will be explained briefly in this section.

2.1.1. The LOLITA Natural Language Processing System

LOLITA [2,3] (Large-scale, Object-based, Linguistic-Interactor, Translator, and Analyser) is a general purpose NLP system that has been under development at the University of Durham since 1986. The aim is to develop a core platform that can be used in many applications. In that platform, two main operations are provided: converting the input text to a logical representation of the meaning contained in it, and retrieving the text from this representation. Some applications are also developed on this core platform like information extraction and simple meaning-based translation. Beside the other core parts and supplementary applications, the most important part of the core is the large knowledge-base called the “Semantic Network”. This is used in most of the stages in the analysis of the input and the representation of the input is added to it afterwards. Figure 2.1 shows a piece of the semantic network for the sentence “Ahmet ate an apple”.

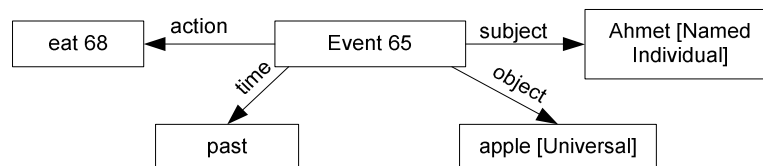


Figure 2.1. Example piece of semantic network from LOLITA [2]

This network is a directed graph. Each node has a name and represents either an entity or an event. Each node includes links (relations between nodes) and controls. Each link includes an arc, which is also a node. Using a node structure for an arc provides representing different kinds of possible relationships with the same structure. Controls give basic information about the node like syntactic category (noun, adverb, etc.), class (human, animal, etc.), type (event or entity) and quantification information (universal, individual). Approximately 60 different types of arcs (subject, object, known fact, hypothesis, etc.) are used in the system and subject, object and action are accepted as the basic roles of an event.

The analysis of the input starts with format conversion. The initial format of the input, the format of the application where the input is used, must be converted to a tree where each constituent word is represented by a node. Some morphological processes (like contradiction unpacking, suffix stripping) are executed on the tree before sending it to the parser. After the whole sentence is parsed, a “parse forest” is retrieved and the best parse tree is selected from this forest. Finally, this tree is converted to a piece of semantic network like in Figure 2.1 using the semantic and pragmatic stages.

2.1.2. GALAXY

GALAXY [4] is a system with which users can communicate either by using a keyboard or by using a microphone. The language accepted by the system is English. It has a client/server architecture and different servers (divided into the “technology” and “domain” server categories), are used for generating answers to the queries. The technology servers are used for different purposes: SUMMIT (speech recognition), TINA (language understanding), and GENESIS (language generation). The domain servers are used for delivering application specific knowledge: JUPITER (weather forecasts), PEGASUS (airline scheduling), VOYAGER (city guide), DINEX (restaurant guide), WHEELS (online automobile classified ads), and WebGALAXY (selected Web-based information).

A user can receive either a spoken or a displayed answer to his/her query. This system can be launched to the Internet, so more users can access the system. It is planned to extend the system to accept inputs from user gestures and handwriting.

From our point of view, the servers that we are interested in are TINA and GENESIS. The sentence *"Where is the library near Central Square?"* is an example query that can be given to TINA. This server finds the components of the sentence like the verb and the subject of the sentence and formats them in a semantic frame. It is a structure where these components can be stored. The frame retrieved from the parsing of this input sentence is given in Figure 2.2.

Clause: LOCATE
Topic: PUBLIC-BUILDING
Quantifier: DEF
Name: library
Predicate: NEAR
Topic: SQUARE
Name: Central

Figure 2.2. Semantic frame example from GALAXY [4]

GENESIS forms a SQL query using this frame and the related domain server, VOYAGER in this case, submits this query to its database. The results of the query are placed in a semantic frame by TINA. GENESIS extracts and puts the components of this frame into sentences. The result of the query is *"The library near Central Square is located on Massachusetts Avenue between Green Street and State Street."* If chosen, the result is transformed to speech by a speech synthesizer.

2.1.3. TRIPS: An Integrated Intelligent Problem-Solving Assistant

TRIPS, the Rochester Interactive Planning System [5,6], is an assistant to a human manager and it constructs plans with the manager in crisis situations. It participates in dialogues with the user for solving problems in different domains. The components of the system share three areas of functionality: interpretation, generation and behavior.

The managers used in the system are:

- The Interpretation Manager: interprets the input and identifies the discourse obligations with the problem solving acts that the user attempts to accomplish.
- The Reference Manager: identifies the likely referents of the referring expressions. It uses the discourse context of the preceding sentences and the knowledge of the particular situation.
- The Task Manager: provides task specific information.
- The Generation Manager: generates sentences of the system and outputs them.
- The Behavioral Agent: is the main module of the system and is responsible for the overall behavior of the system. The selection of the proposed task is done by this agent.

The Discourse Context is a part of this system and includes information about the discourse structure of the conversation.

The inputs are retrieved from the speech recognizer or from the keyboard. The input is sent to a parser. Using the syntactic and semantic information, a sequence of speech acts [5,7] and the predicates defined in the scenario files are retrieved from the parser. The output of the sentence “We need to get the woman in Penfield to Strong” is given in Figure 2.3.

```
(ASSERT
:ID SA11
:SPEAKER USR
:HEARER SYS
:CONTENT
(NEED
:AGENT (PRO “we”)
:THEME
(TRANSPORT
:OBJECT
(THE ?w (AND
(TYPE ?w WOMAN)
(AT-LOC ?w
(NAME ?n “Penfield”))))
:TO-LOC (NAME ?s “Strong”))))
```

Figure 2.3. Output of the parser of TRIPS [5]

This output is sent to the interpretation manager. It refers to the task manager, the reference manager and the discourse context in order to determine what the problem act of the system will be. The resolved problem act is sent to the behavioral agent and it decides the response of the system. Finally, this response is sent to the generation manager and it communicates with the user.

2.1.4. TOY

Çetinoğlu [1] developed a natural language infrastructure for processing Turkish sentences in her M.S. thesis. TOY (“Türkçe Okur Yazar”- “Literate (literally, “reads and writes”) in Turkish”), which is a prototype man-machine communication program, is the main application of her work. The design and implementation of this infrastructure are open to further additions and modifications. So, the infrastructure can be effectively used in constructing various NLP applications like conversational systems. It forms the basis of Öğün’s [8] and Şeker’s [9] works (see Sections 2.1.4.1 and 2.1.4.2), as well as our own conversational agent.

In TOY, processing of the sentences starts with morphological analysis and continues with syntactic and semantic analysis. Finally, the information in the sentence is asserted to the knowledge base if it is a statement. If the user asks a question, the program forms an answer to this question based on what it already knows. As a result, TOY can be seen as an application that enables the user to ask questions and to give new information to the program. Like many NLP applications, its capabilities are limited and the scenario given in Figure 2.4 gives an idea about what can be done with this application. Prolog is used as the implementation language of this application.

The TOY infrastructure forms the basis of our agent program. We made some modifications on existing parts of the infrastructure (see Chapters 4, 5, and 6); in the lexicon (like changing the structures of the morpheme definitions), in the morphological level (like adding new arcs on the FSM’s), in the syntactic level (like adding new DCG rules) and in the semantic level (like adding new modification rules). New parts like the semantic network hierarchy and the sentence generation module have been added to the infrastructure (see Sections 3.4, 6.3). Our agent has all the capabilities of TOY except

answering wh-questions and processing time units and numbers [1]. Technical details of the TOY infrastructure, as well as the improvements that we incorporated to it, will be explained in the following chapters.

Canan uyudu mu (Did Canan fall asleep?)

Bilmiyorum (I don't know)

Kemal uyudu mu? (Did Kemal fall asleep?)

Evet, Kemal uyudu (Yes, Kemal fell asleep)

Canan küçük bir çocuktur. Kemal küçük bir çocuktur. Bütün küçük çocuklar 10 saat uyurlar. (Canan is a little child. Kemal is a little child. All little children sleep for 10 hours.)

Kemal ne zaman uyudu? (When did Kemal fall asleep?)

Kemal yirmiüçte uyudu (Kemal fell asleep at twentythree hours.)

Kemal ne zaman uyandı? (When did Kemal wake up?)

Kemal dokuzda uyandı (Kemal woke up at nine hours.)

Canan kahvaltıda ne yiyecek?(What will Canan eat at breakfast?)

Bilmiyorum (I don't know.)

Canan kahvaltıda peynir, ekmek, zeytin yiyecek (Canan will eat some cheese, bread, olives at breakfast.)

Teşekkürler öğrendim (Thanks, I learned.)

Küçük çocuklar hariç herkes kahvaltıda çay içer (Everybody except little children drink tea at breakfast.)

Canan kahvaltıda ne içecek? (What will Canan drink at breakfast?)

Bilmiyorum (I don't know.)

Canan kahvaltıda çay içecek mi? (Will Canan drink tea at breakfast?)

Bilmiyorum (I don't know.)

Kaç kişi kahvaltıda çay içmeyecek? (How many people will not drink tea at breakfast?)

Bilmiyorum (I don't know.)

Kim kahvaltıda çay içmeyecek? (Who will not drink tea at breakfast?)

Bilmiyorum (I don't know.)

Kaç kişi kahvaltıda peynir, ekmek, zeytin yiyecek? (How many people will eat some cheese, bread, olive at breakfast?)

Bir kişi kahvaltıda peynir, zeytin, ekmek yiyecek (One person will eat some cheese, bread, olive at breakfast.)

Kim kahvaltıda peynir,ekmek,zeytin yiyecek? (Who will eat some cheese, bread, olive at breakfast?)

Canan kahvaltıda peynir,ekmek,zeytin yiyecek. (Canan will eat some cheese, bread, olive at breakfast)

Figure 2.4. Target scenario of TOY [1]

2.1.4.1. Conversation System Infrastructure for the Turkish Language. Ögün [8] developed a conversation system infrastructure for the Turkish language which was based on the TOY NLP infrastructure [1]. He made some improvements on the morphological, syntactic and semantic levels of TOY and developed some new parts in order to implement the requirements of the proposed system like sentence generation, correspondent representation and additional lexical entries and grammatical rules.

Ögün's architecture consists of two basic parts: the sentence processing mechanism and the agent-based conversation policy. The sentence processing mechanism is responsible for morphological and syntactic parsing of the input sentence. The conversation policy is responsible for the execution of procedures to manage the conversation and for generating respective answers or questions according to the user inputs. This policy structure supports customized knowledge base manipulation, user-specific knowledge maintenance and question generation [8].

The system can participate in a conversation with only one user at a time but can manage series of conversations with different users consecutively. Finding contradictions between the user inputs is one of the major points in this work. Question generation is an important aspect of the system, because it facilitates certain elements of a conversation. The system can decide to ask questions to the user according to specific programmable conversation policies. A conversation example is given in Figure 2.5.

<p>Merhaba ben Canan (Hello, this is Canan) Selam (Hi)</p> <p>Mehmet okula gitmedi (Mehmet did not go to the school) Anladım (Understood)</p> <p>Merhaba ben Mehmet (Hello, this is Mehmet) Merhaba (Hello)</p> <p>Ben okula gittim (I went to the school) Anladım (Understood)</p> <p>Canan diyor ki Mehmet okula gitmedi (Canan said that Mehmet did not go to the school) Emin misin? (Are you sure?)</p> <p>Hayır (No)</p>
--

Figure 2.5. A conversation example from Ögün's conversation system [8]

2.1.4.2. Turkish Speaking Assistant (TuSA). Şeker [9] developed a Turkish speaking assistant for storing and retrieving information about appointments for people who do not have familiarity with computers. The user enters the appointment details using Turkish sentences. TuSA is also based on the TOY infrastructure, but the internal representation of the knowledge is completely different. Some modifications and additions were also applied on the remaining parts of the infrastructure.

Appointments are stored in a file and each entry has the structure

takvim(ID,Minute,Hour,Day,Month,Year,Person,Location,Subject,Duration,Recurring)

where

- ID: the unique number given to the appointment.
- Minute and Hour: the representatives of the time phrases like “onu elli geçe” (at 50 past 10 o’clock).
- Day, Month and Year: the date of the appointment
- Person, Location and Subject: with whom, where and about what the appointment will be. Subject can be a meeting, a dinner, an interview or anything.
- Duration: either minute or hour
- Recurring: recurring event information like “aliyle iki günde bir toplantı var” (there is a meeting with ali every two days). Recursion can be “daily”, “monthly” or “yearly”. The semantic representation used in this slot is

recur(Type, Period, Period of month, Period of day number, Long format of day, Final day, Final month, Final year).

If a sentence like “*Haftaya Aliyle olan yemekli toplantıları göster*” is given as an input, TuSA converts the query to a internal formula like *takvim(,_,_,22,11,2002,_,,[ali],[yemekli,toplantı],_,_)* and runs this query on the database. The results of the query, if found, are shown to the user at the end in the format shown in Figure 2.6.

<p>ID:11</p> <p><i>Tarih:22/Kasim/2002 09:58 Cuma</i> (Date:22/November/2002 09:58 Friday)</p> <p><i>Kisi: arkadasim ali</i> (Person: my friend ali)</p> <p><i>Yer:komsumuz Canan</i></p> <p>(Location: Neighbor Canan)</p> <p><i>Konu:yemekli bir toplanti</i> (Subject: Dinner)</p> <p><i>Suresi:7 saat</i> (Duration: 7 hours)</p> <p><u>Semantics:</u></p> <p>takvim(11,9,58,22,11,2002,[komsumuz,canan], [arkadasim,ali],[yemekli,bir,toplantı],7,'saat').</p>

Figure 2.6. Results of a query given to TuSA [9]

2.2. Commonsense Knowledge Representation and Lexical Reference Systems

Humans have access to a great amount of commonsense knowledge, and a computer program that does not possess this same knowledge cannot be expected to perform comparably to humans. If the knowledge item “a person can not fly” is not available to a program, then it may reasonably suppose that an object mentioned to be flying in a sentence can be a human.

The concept of semantic network was proposed by Charles Peirce under the name of “existential graph” [10] in 1909. Semantic networks can be seen as a knowledge representation technique for recording all the relevant relations between members of set of objects and classes. "Object" means an individual like a particular person, a particular animal or a particular thing (book, table, etc.). "Class" means a set of related objects - the set of all persons, all books, all tables, etc. A very important type of possible relation among classes is “subset” (class membership). Any other relation, like “likes” or “sister of”, can also be defined. A semantic network is represented as a directed graph as shown in Figure 2.7.

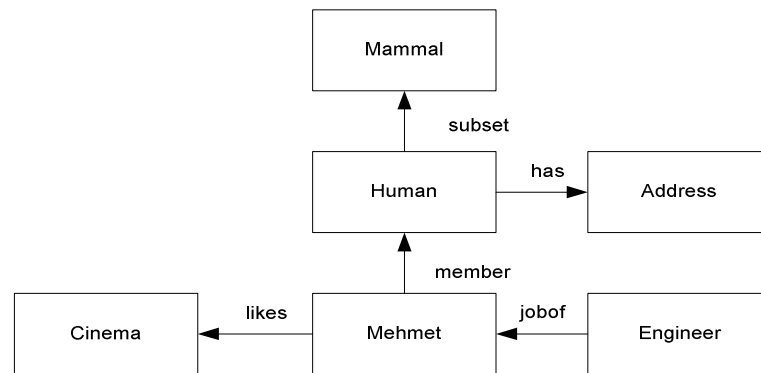


Figure 2.7. Semantic network example

In Figure 2.7, the boxes are the objects and classes, the labels of the arcs are the relations between the objects and classes. A ‘subset’ link shows that the “human” class is a subset of the “mammal” class (in other words “every human is a mammal”) and the ‘jobof’ link shows that Mehmet is an engineer.

Inheritance between classes and their subclasses can be easily represented by using a semantic network. In Figure 2.7, it is shown that Mehmet is a human being; as a result, he inherits all characteristic features of human beings like having an address. Multiple inheritance is possible in semantic networks, which means an object can belong to more than one class and one class may be a subset of more than one class.

In the following subsections, different commonsense knowledge representations in different models and lexical reference systems will be explained briefly.

2.2.1. Cyc

Cyc is presented as the world's largest and most complete general knowledge base and commonsense reasoning engine [11]. OpenCyc is the open source version of the Cyc technology. From the beginning, Cycorp, the builders of Cyc, entered literally millions of logical axioms, rules and other assertions which specify constraints on the individual objects and classes found in the real world into the Cyc.

The Cyc ontology satisfies the following two important criteria:

- **Being universal:** Every concept in nature can be correctly linked into the ontology in appropriate places, no matter how general or specific.
- **Being articulate:** The distinctions in the ontology are both necessary and sufficient for most purposes. It can effectively be used in applications (natural language understanding and generation, semantic database integration, consistency checking, ontology-constrained simulation, etc.) because it supports knowledge sharing, integration and language disambiguation.

Concepts in the knowledge base are represented as Cyc constants, which are also called terms or units. By convention, the name of each constant begins with the characters `#$`; for example, `#$Skin`. A constant can represent a collection (such as the set of all mammals), an individual object (such as a particular animal), a word in a natural language (such as the word 'apple'), a quantifier (such as 'there exists'), a relation (a predicate, function, slot, attribute, etc.), and so on.

The constant “skin” is defined in the ontology as seen in Figure 2.8.

<p><code>#\$Skin</code> (1)</p> <p>A (piece of) skin serves as outer protective and tactile sensory covering for (part of) an animal's body. This is the collection of all pieces of skin. Some examples include <code>#\$TheGoldenFleece</code> (representing an entire skin of an animal) and (<code>#\$BodyPartFn</code> <code>#\$YulBryner</code> <code>#\$Scalp</code>) (representing a small portion of his skin). (2)</p> <p>isa: <code>#\$AnimalBodyPartType</code> (3)</p> <p>genls: <code>#\$BiologicalLivingObject</code> <code>#\$AnimalBodyPart</code> <code>#\$SheetOfSomeStuff</code> <code>#\$VibrationThroughAMediumSensor</code> <code>#\$TactileSensor</code> (4)</p>
--

Figure 2.8. Definition of the constant ‘skin’ in the Cyc ontology [11]

Each definition starts with the name and an English comment explaining the meaning and intended use of the constant (1 and 2). Then some hierarchical information is given like the sets that this constant is an element of, the supersets of the constant, etc (3 and 4). These are explained briefly in [11]. Each constant is an element of one or more collections. So in each definition there is one “isa” line. Each collection to which this constant belongs

is separately written in this line. #Skin is itself a collection (the set of all pieces of skin, including as a subset the set of all 'whole skins'); it can therefore have supersets and subsets and elements. The line starting with "genls" shows some of the supersets of #Skin, such as #TactileSensor and #SheetOfSomeStuff. That means that if x is a piece of skin, then x is a sheet of some stuff, and x is a tactile sensor, and so on. Clutter avoidance is also implemented in the ontology.

Mayfield, Finin, Narayanaswamy, MacCartney and Goolsbey [12] developed a system, the Cyclic Friends Network (CFN), that explores cooperative inference across the Internet where the Cyc knowledge base and inference engine are used as a testbed.

2.2.2. WordNet

WordNet [13] is a lexical reference system where words of different syntactic categories are organized into synonym sets (synsets) and linked by using different semantic relations. The construction of this system is based on human lexical memory. All around the world, lots of researches participate in the developing process of wordnets for different languages like English and Turkish. Using the online versions, anybody can easily get information about that language's word taxonomies and the relations between the words like hyperonym and hyponym of each word.

Different consortiums work cooperatively in order to develop a general WordNet for more than one language. For the development of a general WordNet, initially each language's WordNet is developed according to the general vocabulary of that language. Semantic relations will be classified in the independent WordNets according to a shared ontology. Then, all developed WordNets will be stored in a common database providing linking across them. Using these general WordNets, it is possible to go from a word in one language to similar word(s) in any other language. Euro WordNet is a good example of such a cooperation [14].

The rest of this subsection includes brief introductions to various WordNets in different stages of development.

Princeton Wordnet: This project is developed by the Cognitive Science Laboratory at Princeton University under the direction of Professor George A. Miller [13]. In this system, English nouns, verbs, adjectives and adverbs are organized into synonym sets and these sets are linked by using different relations.

Euro WordNet Project: The aim of this project is to develop a lexical database with WordNets for several European languages (Dutch, Italian, Spanish, German, French, Czech and Estonian) [14]. These WordNets are structured in the same way as the Princeton WordNet and linked to an index which shows the semantic relations between them. This project was completed in 1999. Figure 2.9 shows a part of a data sample from this WordNet where the literal “able” is explained. The cooperative framework of EuroWordNet is continued through the Global WordNet Association.

```

WORD_MEANING
  1 PART_OF_SPEECH "a"
  1 VARIANTS
    2 LITERAL "able"
      3 SENSE 1
        3 DEFINITION "having the necessary means or skill or know-how or authority; esp having become able;
usu followed by "to": "able to swim"; "she was able to program her computer"; "we were at last able to buy a
car"; "able to get a grant for the project""
      3 EXTERNAL_INFO
        4 SOURCE_ID 1
          5 TEXT_KEY 00002403-a
    2 LITERAL "capable"
      3 SENSE 1
        3 EXTERNAL_INFO
          4 SOURCE_ID 1
            5 TEXT_KEY 00002403-a
.....

```

Figure 2.9. Data sample from Euro WordNet [14]

Balkan WordNet (BalkaNet) Project: This project aims to develop a multilingual lexical database comprising of individual WordNets for the Balkan languages [15]. It attempts to

represent semantic relations between words in each Balkan language and link them together for developing an on line multilingual semantic network.

Turkish WordNet Project: This project [16] is a part of BalkaNet Project [15] and focuses on the construction of a Turkish lexical database. It is under development at Sabancı University.

2.2.3. Role Assignments Model

McClelland and Kawamoto [17] propose a distributed model to assign roles to constituents of sentences. In this model, the assignment mechanism uses semantic constraints. The model is able to learn based on its experience with sentences. Beside the main goal, there are other goals in the model:

- The model is aimed to select contextually appropriate readings of ambiguous words.
- The model is aimed to select the appropriate verb frame based on the pattern or arguments and their semantic features.
- The model is aimed to fill in missing arguments in incomplete sentences with plausible default values.
- The model is aimed to generalize its knowledge of correct role assignment to sentences containing a word it has never seen before, given only a specification of some of the semantic properties of the word.

In this model, words are represented as lists of semantic microfeatures. For both nouns and verbs, the features are grouped into several dimensions. Feature dimensions and values of nouns and verbs are given in Table 2.1 and Table 2.2 respectively. The feature details are explained in [17].

Table 2.1. Feature dimensions and values of nouns from role assignment model [17]

Human	Human nonhuman
Softness	soft hard
Gender	male female neuter
Volume	small medium large
Form	compact 1-D 2-D 3-D
Pointiness	pointed rounded
Breakability	Fragile unbreakable
Obj-Tpe	food toy tool utensil furniture animate natural inanimate

Table 2.2. Feature dimensions and values of verbs from role assignment model [17]

Doer	yes no
Cause	yes no-cause no-change
Touch	agent inst both none Agent_ is_ patient
Nat-Chng	Pieces shreds chemical none unused
Agt-Mvmt	trans part none Not_applicable
Pt-Mvmt	trans part none Not_applicable
Intensity	low high

Each dimension consists of a set of mutually exclusive values, and each word is represented by a vector in which one value on each dimension is 1 for the word and all of the other values are 0. Table 2.3 shows the vector of the word “ball” and Table 2.4 shows the vectors of the verb “broke”. There can be more than one vector for a verb, in order to represent alternative feature patterns.

Table 2.3. Vector of the word “ball” from role assignment model [17]

	HU	SO	GND	VOL	FORM	PO	BR	OBJ-TYP
ball	01	10	001	100	1000	01	01	0100000

Table 2.4. Vectors of the verb “broke” from role assignment model [17]

	DO	CAU	TOUCH	N_CHG	A_MV	P_MV	IN
Broke	10	100	01000	10000	0100	0010	01
broke (agent-verb-patient)	10	100	10000	10000	0100	0010	01

2.3. Anaphora Resolution Methods

Anaphora is the use of pronouns or other syntactic entities (anaphors) to refer to people, places, or things previously mentioned [18]:

“Canan yemek yedi ise o doymuştur” (If Canan has eaten, she must be full): “o” (she) is an anaphor and refers to Canan.

While a person is talking / writing repeatedly about the same thing, place, or person during a discourse, that entity is frequently referred with an anaphor. The hearer or the reader, already knowing the mentioned things, matches all anaphors with their referents. This matching process is known as resolving anaphoric reference [18]. In resolving an anaphoric reference, the sentence where it is used and the context where this sentence belongs play an important role. The context helps to draw a mental picture about what is going on, and the sentence points to the features that the referent must have. Generally, this is the resolving method used by human beings who already have a commonsense knowledge base.

In a discourse, the anaphora may succeed or precede its referent:

“Canan kahvaltıda süt içmeli ise o bir çocuktur.” (If Canan must drink milk at breakfast, she is a child)

“O bir çocuk olduğu için, Canan kahvaltıda süt içmeli.” (She is a child, so that Canan must drink milk at breakfast).

In the first example, previously mentioned entities are examined for searching the referent. But in the second one, later ones must be examined. The latter is called backwards anaphora or cataphora, and this requires the pronoun and the referent to be in the same sentence [18].

Anaphora resolution is regarded to be a hard problem in NLP. It requires the integrated application of syntactic, semantic, and pragmatic knowledge [19]. This is a field of ongoing research, and several scientists have developed different resolution techniques which address different subsets of the problem.

The traditional methods [20,21,22] are heavily based on linguistic and domain knowledge. The construction of such a knowledge base system is a time-consuming and labour intensive task [23]. Syntactic, semantic and discourse analysis play important roles in these methods and many factors are taken into account during the resolution process. The types of these factors, their priorities, and the implementation form the differences between these methods. Some of these factors are given in Table 5.4.

Table 2.5. Important factors in anaphora resolution

Factor	Description
Number	The number of the anaphor and the referent must agree
Gender	The gender of the anaphor and the referent must agree
Postconditions versus Preconditions	Postconditions of the actions associated with the referent must not violate the preconditions associated with the anaphor
Distance between the anaphor and the referent	The referent is probably close to the anaphor in the discourse
Pragmatic knowledge	Restriction of the referents based on the knowledge about the external world

Mitkov, Belguith and Stys [23] developed a robust, knowledge-poor method which is promoted to be a multilingual method that can easily be adapted to any language with high success rates. This approach refuses to use traditional linguistic methods and domain

knowledge, which require human input and computational expense. This method is a combination of sentence segmentation, speech tagging, noun phrase identification, and antecedent indicators. It sends the sentence containing the anaphor and three preceding sentences to sentence segmentation, speech tagging and noun phrase identification processes. The output of this operation is a list of filtered candidates on the basis of gender and number agreement. The candidates in this list are checked against the antecedent indicators (like definiteness, givenness, indicating verbs, etc.) and each candidate has a score at the end. The one with the highest score is selected as the referent. A selection method between candidates with the same score is also implemented.

The method does not incorporate syntactic and semantic knowledge and it is expected to have a performance worse than the methods that use this knowledge. It has a success rate of 89.7 % on the average when applied on some English discourses.

Tin and Akman [20] developed an approach to pronominal anaphora resolution in a computational framework (BABY-SIT) employing situation-theoretic constructs. Turkish natural language sentences are used in BABY-SIT. This medium is based on situation theory and the world is accepted as a collection of objects like individuals, places, situations, relations, etc. The situation of each sentence is a combination of its constituents' situations. Each situation description represents its own restrictions, and using these and defined inference rules, pronominal anaphors are resolved.

In the statistical method developed by Ge, Hale, and Charniak [21], an antecedent function is used. This function is derived from some anaphora resolution factors: the distance between the pronoun and the proposed antecedent, gender/number of the proposed antecedent, and noun phrase repetition. One of the proposed antecedents of the pronoun that has the highest value is accepted as the antecedent. One of the main characteristics of this method is its essential simplicity.

Kennedy and Boguraev [22] propose an algorithm that has two main constituents: discourse referents and coreference classes. Each noun phrase and third person pronoun is represented as a discourse referent. A coreference class contains the discourse referents which are believed to be equivalent. Each coreference class has a salience value which is

evaluated after each new referent addition. Salience factors of classes and referents have their own weights in this computation. In order to resolve a pronoun, a list containing candidates is created. The candidate with the highest salience value is accepted as the antecedent of this anaphor.

Carbonell [19] notes that syntactic information is most important in cases where the anaphor and its referent are in the same sentence. Semantic and pragmatic information are most important in other cases where they are only in the same discourse, not in the same sentence. So, Carbonell developed a method using a combination of a set of strategies rather than a single strategy. None of the strategies used in this method are based on mathematical functions or statistics. Some of the strategies that are defined in his work are:

- Local anaphor constraints (number, case, gender constraints that the anaphor carries):

John and Mary went shopping. He bought a steak [he=John]

- Case-role semantic constraints (the constraints that must be satisfied by the referent):

John took the cake from the table and ate it. [it=cake]

- Precondition/postcondition constraints (the postconditions associated with the referent must not violate the preconditions associated with the anaphor):

John gave Tom an apple. He ate the apple. [he=Tom]

- Case-role persistence preference (find the referents in the antecedent phrases that occur in the same semantic case role as the anaphor): The most appropriate candidate found after applying all these strategies is accepted as the antecedent of the anaphor.

Mary gave an apple to Susan. John also gave her an orange. [her=Susan].

3. OVERVIEW OF TOYagent

In this chapter, we give an overall picture of TOYagent, which is the program developed during this thesis work. The two main operations performed by this program are obtaining knowledge and answering questions about the knowledge it has. The modules shown in Figure 3.1 perform different parts of these operations.

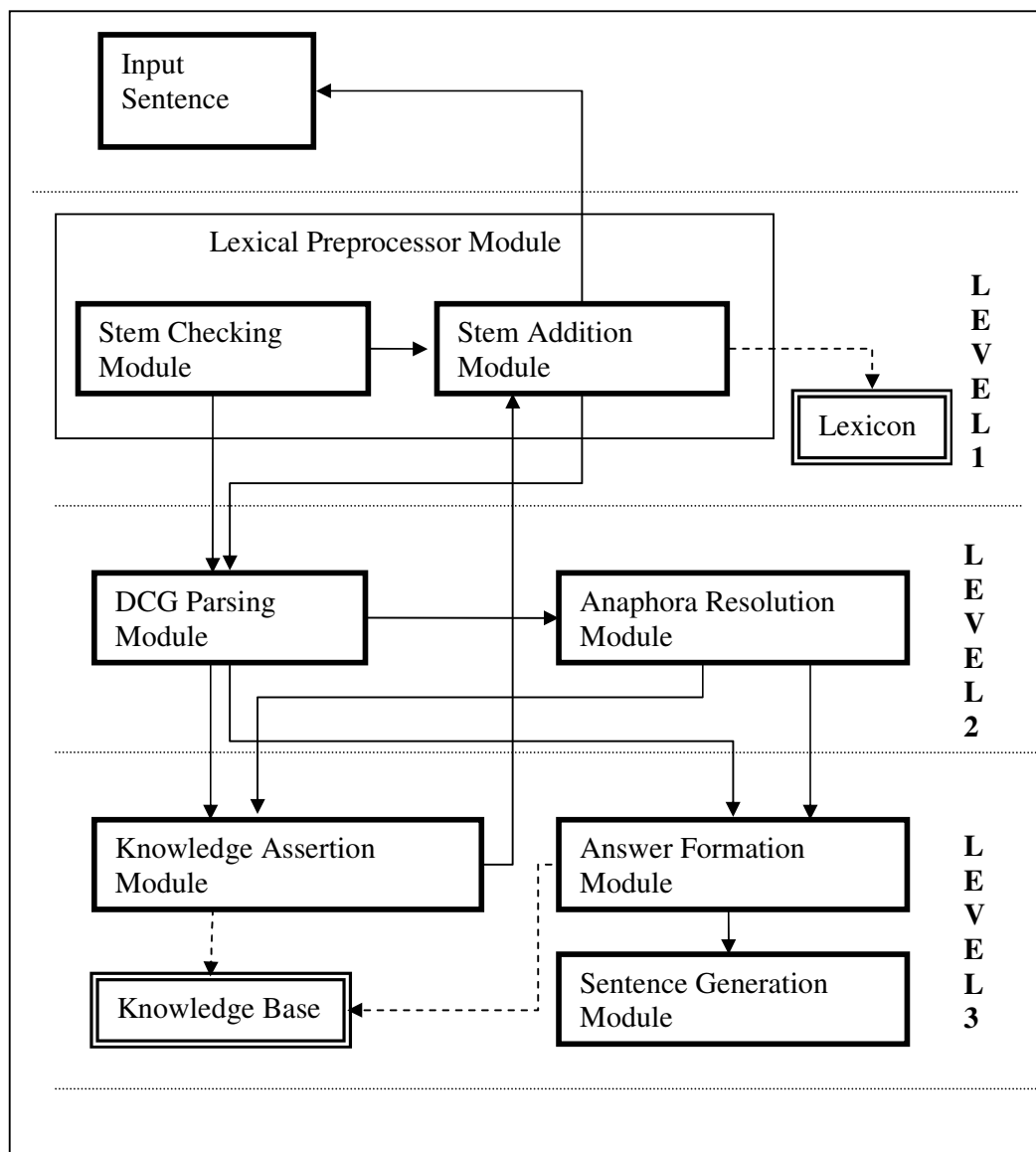


Figure 3.1. General schema of TOYagent

If an input sentence is given to the agent, the execution correctly ends either at the knowledge assertion module (when the agent learns something), or at the answer formation module (when the agent answers a question), or at the sentence generation module (when the agent answers a question and explains its reasons).

3.1. Definitions of Levels and Modules

When an input sentence is given to the agent, it is processed in three levels: the morphological level, the syntactic/semantic level and the transformation/generation level. The knowledge base and the lexicon are accessed and updated during these levels. Although all modules will be explained in detail in the following chapters, their brief definitions are given below:

Stem Checking Module: This module is responsible for checking whether each word used in the input sentence has an entry in the lexicon. If any words that have no entries in the lexicon are found, then the agent executes the stem addition module. Otherwise the agent continues with the DCG parsing module. This module will be explained in Section 5.1.1.

Stem Addition Module: This module is responsible for adding entries to the lexicon for the words found in the stem checking module. User interaction is needed in this module. If a word is added to the lexicon, the user also gives a description of the meaning of this word to the agent. This module will be explained in Section 5.1.2.

These two modules are new to TOYagent. In TOY, it is assumed that all words in the input sentence already have their entries in the lexicon. At the end of the first level, each word in the input sentence is morphologically checked.

DCG Parsing Module: This module is responsible for constructing the semantic formula (see Section 3.3) of the input sentence by using the defined DCG rules. If the input sentence is a conjoined sentence, then one formula for each conjunct sentence is constructed. First order predicate calculus (FOPC), which will be explained in Section 3.2,

is used in the representation of semantic structures and relations. This module will be explained in Section 5.2.

Anaphora Resolution Module: If an anaphor is used in the conjoined sentence given as input, this module finds its referent and replaces the anaphor with that referent. At the end, a new semantic formula is obtained. If a referent is not found, the semantic formula of the sentence remains the same. A semantic network hierarchy (see Section 3.4) is used for the resolution mechanism. The resolution module will be explained in Section 5.3.3.

At the end of the second level, the sentence has been syntactically and semantically checked. We made some additions and modifications to TOY's DCG rules. The anaphora resolution module is new to this work.

Knowledge Assertion Module: This module is responsible for asserting the knowledge contained in the input sentence if the sentence is not a yes/no question. The operations performed in this module vary according to the semantic formula of the sentence and any conjunctions that may appear in the sentence. This module will be explained in Section 5.4.

Answer Formation Module: This module is responsible for forming answers to yes/no questions given by the user. Three mechanisms are used to form the answer. In one of these mechanisms, the semantic network hierarchy that we developed is used. If an answer can not be generated, the agent outputs "Bilmiyorum" (I don't know), otherwise, it outputs the formed answer. This module will be explained in Sections 6.1 and 6.2.

Sentence Generation Module: This module is responsible for explaining how the answer of the given yes/no question is formed by the agent. The outputs of this module are Turkish sentences. The information used in the answer formation module is retrieved and expressed in this module. The agent has two modes (explanatory and brief), and if it is in explanatory mode, this module will be executed automatically after each answer formation. This module will be explained in Section 6.3.

At the third level, what will be done with the input sentence is determined. This level is focused on sentence understanding.

The next three sections describe the meaning representation employed by the program. The final section of this chapter contains examples for the processing of sentences by TOYagent.

3.2. First Order Predicate Calculus (FOPC)

First Order Predicate Calculus (FOPC) and its extensions are used in many NLP applications [1,8,9], to represent semantic structures and relations. FOPC is selected because it is a quite expressive language which is also easy to understand. Representation of a sentence using FOPC also eases its transformation to Prolog and this is another reason for choosing FOPC as an intermediate representation. Constants, predicates, variables, connectives and quantifiers are used in FOPC expressions. (Table 3.1)

Table 3.1. FOPC constituents and their uses in language processing

Constants	Proper nouns or personal pronouns	Canan , I
Predicates	Common nouns, adjectives, verbs	mother, angel, have
Variables	Representatives (not names) of entities that have a given property	Y
Connectives	Symbols representing a relation among sentences	\rightarrow (implies), \wedge (and)
Quantifiers	Scopes of variables	\forall (Universal), \exists (Existential)

Using the constituents shown in Table 3.1, in FOPC the sentence “Bütün anneler güzeldirler ve melektirler” (All mothers are beautiful and angels) is represented as

$$(\forall X) \text{mother}(X) \rightarrow \text{beautiful}(X) \wedge \text{angel}(X)$$

and the sentence “Canan’ın bebeği vardır” (Canan has a baby) is represented as

$$(\exists Y) \text{baby}(Y) \wedge \text{have}(\text{Canan}, Y)$$

More than one representation can be used to talk about the world in FOPC. The number of arguments in the predicates and predicate names used in these representations make the difference:

“Canan ekmek yedi” (Canan ate some bread)

$(\exists X) \text{bread}(X) \wedge \text{eat}(\text{Canan}, X)$

$(\exists X) \text{bread}(X) \wedge \text{do_action}(\text{Canan}, X, \text{eat})$

In FOPC, being something, like “being beautiful”, or “being a mother”, etc. is represented as a property of the thing mentioned. In the above example, “Canan ate some bread” is not represented as $\text{eat}(\text{Canan}, \text{bread})$ because, being a (piece of) bread is used as a property of the quantified variable X . To represent such properties independently of the things that they belong to, lambda calculus [1,8,9,18] is used. The thing is represented as the argument of the property predicate.

The property of being (a piece of) bread, that is, the contribution of the word “bread” to the semantic formula of any sentence that it appears in, is expressed in lambda calculus as

$$\lambda x \text{bread}(x).$$

Lambda expressions are denoted by using ‘ \wedge ’ as an infix operator in Prolog, so this same property is expressed in Prolog as $X^{\wedge}\text{bread}(X)$.

In FOPC, quantifiers, most notably, the universal (\forall) and existential (\exists) quantifiers, are used to indicate the scopes of the expressions that they precede. A universally quantified expression is true if and only if all the instances of the expression are true. An existentially quantified expression is true if and only if at least one instance of the expression is true [1]. FOPC expressions are written in Prolog notation in TOY. This is performed by defining individual semantic subformulas of all syntactic categories and the quantifiers.

FOPC notation

$(\forall X) \text{mother}(X) \rightarrow \text{beautiful}(X) \wedge \text{angel}(X)$
 $(\exists Y) \text{baby}(Y) \wedge \text{has}(\text{Canan}, Y)$

Prolog Notation

$\text{all}(X, \text{mother}(X), (\text{beautiful}(X), \text{angel}(X)))$
 $\text{some}(X, \text{baby}(X), \text{has}(\text{'Canan'}, X))$

In Prolog notation, the second argument is the restrictor and the third argument is the scope of the quantifier. The restrictor defines a subset of possible X values that can be taken by the quantified variable. The described X values are the ones that make both the restrictor and the scope true. In the first FOPC formula above, the set of all mothers is visualized, and any member of this set is said to be both beautiful and an angel.

3.3. Semantic Formula of a Sentence

This formula is basically a (possibly nested) FOPC expression corresponding to a complete sentence in Prolog. This expression is constructed during the DCG parsing of a sentence from smaller subformulas corresponding to the syntactic constituents, and is produced at the end of the DCG parsing module in TOYagent. In our acceptance, the meaning of a sentence is obtained by combining the meanings of its constituents, although there are other factors that must be taken into account. This expression is the basic intermediate representation used between levels two and three in Figure 3.1. This formula is used in order to assert the knowledge contained in the input sentence or to query the knowledge. Details of this formula and the operations performed on it will be explained in Sections 5.2, 5.3, and 5.4.

The semantic formula of the sentence “*çocuk süt içti*” (The child drank milk), as well as an indication of the syntactic components giving rise to its subparts, are given in Figure 3.2.

As will be seen in Section 5.2, the parsing of each “atomic” sentence starts from the $S \rightarrow NP VP$ syntax rule. If a noun (property) is encountered in the sentence, an appropriately quantified expression for this is added to the FOPC expression of the sentence. In this quantified expression, the property constructs the restrictor.

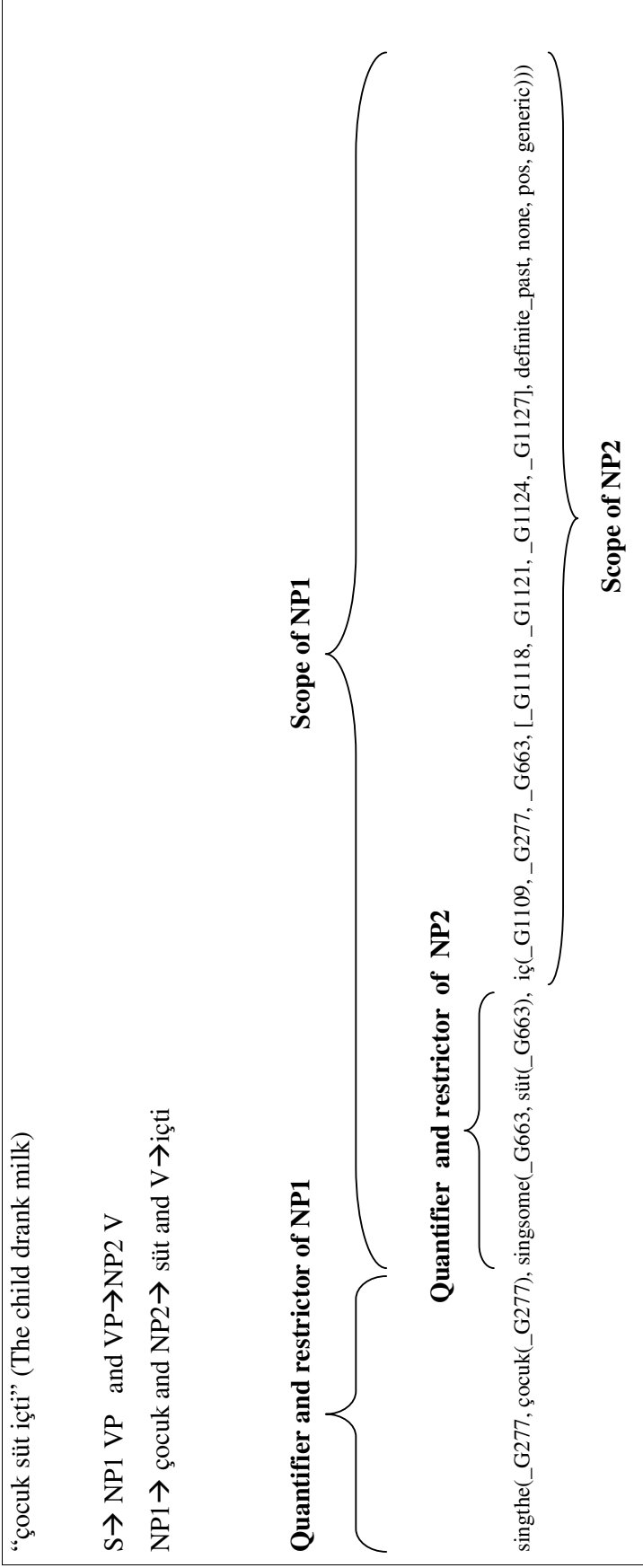


Figure 3.2. Example semantic formula

Quantifiers in a formal logic representation are counterparts of determiners in natural language. Each determiner in a sentence corresponds to a quantifier, and these play an important part in the semantic formula, as will be explained in Section 5.2.3.1.

3.4. Semantic Network Hierarchy

We incorporated a “homegrown” semantic network to serve as an extendible commonsense knowledge repository to TOYagent.

In order to construct the network hierarchy, we determine the most general groups for classifying objects like humans, animals, abstract entities, etc. The name of each group indicates its characteristics.

Figure 3.3 shows the semantic network implemented to characterize a subset of the noun entries:

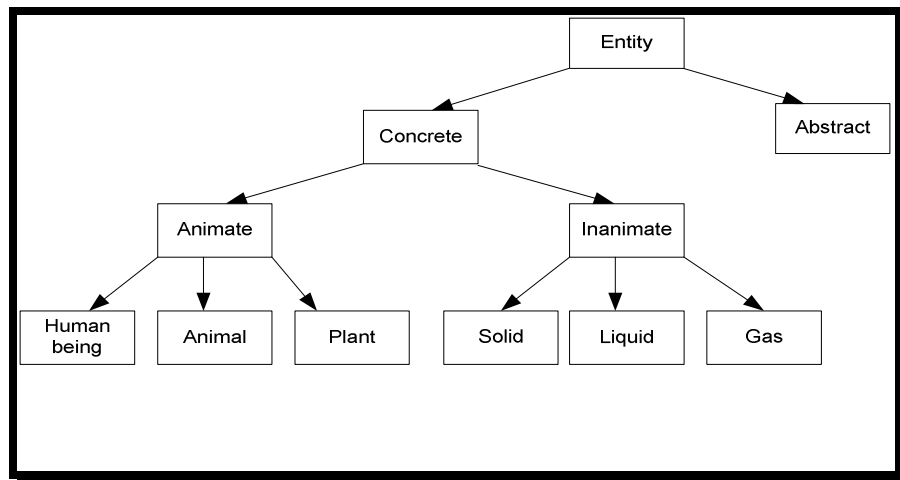


Figure 3.3. Semantic network hierarchy

This is really a small part of the semantic network that can be generated for a detailed representation of all possible groups in nature. Each node represents a class that the objects can belong to and each arc represents the same (“subset”) relation between the classes. Entity is the most general class to which an object belongs to. Concrete and Abstract classes are subsets of the Entity class. The semantic class of each word is

represented by a path from root to the node it belongs in TOYagent (from the most general class to the least general one). The structure of this list is explained in Section 4.2.2. Objects that belong to the same node are assumed to have semantic similarities.

This knowledge is currently used by the anaphora resolution module and by one of the question answering mechanisms:

If the sentence, “Kemal yemeği pişirdi ise o lezzetlidir” (If Kemal cooked the meal, it is delicious) is given, the agent uses this hierarchy to find the possible referents of the word “o” (it) and finally obtains a representation corresponding to “Kemal yemeği pişirdi ise yemek lezzetlidir” (If Kemal cooked the meal, the meal is delicious). Details of this resolution process will be explained in Section 5.3.3.

If “Ayşe kahvaltıda çay içti” (Ayşe drank tea at breakfast) was asserted to the knowledge base earlier, the agent has learned that Ayşe drinks tea, a hot beverage, at breakfast. If “Ayşe kahvaltıda kahve içer mi?” (Can Ayşe drink coffee at breakfast?) is given to the agent, it recognizes the semantic similarities of coffee and tea and deduces that Ayşe can possibly drink coffee at breakfast. This hierarchy is used in the determination of semantic similarities between two entities, which are tea and coffee in this case. Details of this deduction mechanism will be explained in Section 6.2.

3.5. Processing of Sentences by TOYagent

Figure 3.4 and 3.5 show a dialogue example where the user gives the sentences “Ali Kaşığı okudu” (Ali read the kaşığı. (This is the title of a book.)) and then “Kemal kitap okuyabilir mi?” (Can Kemal read a book?) as inputs to the agent. (Orthographic hints, like quotes or capital letters, that are useful for understanding that we are talking about a book title, are not available here, much like the situation faced by the hearer in spoken discourse.) Figure 3.6 shows the processing of the first sentence and Figure 3.7 shows the processing of the question if the agent is in brief mode. The numbers used in the figures show the order of the operations performed.

Türkçe Okur Yazar (Literate in Turkish)

Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)

Cümleniz> **Ali kaşığı okudu. (Ali read the kaşığı) [1]**

T_agent:kaşığı kelimesini bilmiyorum (I don't know the meaning of the word "kaşığı") **[2, 3, 4]**

Kelimenin Kökü hangisi: (Which one is the root of the word?) **[4]**

1 : kaşı

2 : kaşığı

3 : kaşığı

Hangisi (Which one): **1.**

T_agent: kaşığı kelimesinin türü nedir ? (What is the type of the word "kaşığı") **[4]**

1:İsim (Noun)

2:Sıfat (Adjective)

3:Zaman (Time_unit)

4:Özel İsim (Propernoun)

Hangisi (Which one):**1.**

T_agent: "kaşığı kelimesinin tanımıyla ilgili sorular sorayım mı?(e/h)

l: **e. (yes)**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:

somut soyut

> **somut. (concrete)**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:

canlı cansız

> **cansız. (inanimate)**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:

katı sıvı gaz

> **katı. (solid) [4]**

Figure 3.4. Dialogue example

tr_morph_entry(AdKök, [[k, a, ş, a, ğ, ı], [type(noun), sem(_G1476^kaşığı(_G1476))], [sm,cns,kt]], _, _, _, l, l, ok) **5, 6**

T_agent:kaşığı ne demek? Anlatır mısın? (What does kaşığı mean? Can you explain?)
l: kaşığı bir kitaptır. (kaşığı is a book) 7, 8, 9, 10

semantic formula:singthe(_G2554, kaşığı(_G2554), kitap(_G2554)) **11**
 kitap(_G2554):-kaşığı(_G2554). **12, 13**

T_agent:Teşekkürler, öğrendim... (Thanks, I learned).

semantic formula:singthe(_G3827, kaşığı(_G3827), oku(_G4300, ali, _G3827, [_G4309, _G4312, _G4315, _G4318], definite_past, none, pos, generic)) **15**
 kaşığı(x(1)). **16,17**
 id_table(x(1), _G4755^kaşığı(_G4755)). **16,17**
 oku(_G4300, ali, x(1), [_G4309, _G4312, _G4315, _G4318], definite_past, none, pos, generic). **16,17**

T_agent:Teşekkürler, öğrendim... (Thanks, I learned).

Cümleliz> **Kemal kitap okuyabilir mi. (Can Kemal read a book?) 18,19,20,21**
 T_agent: Mümkündür (-) (Possibly) **22, 23, 24**

Figure 3.5. Dialogue example (continued)

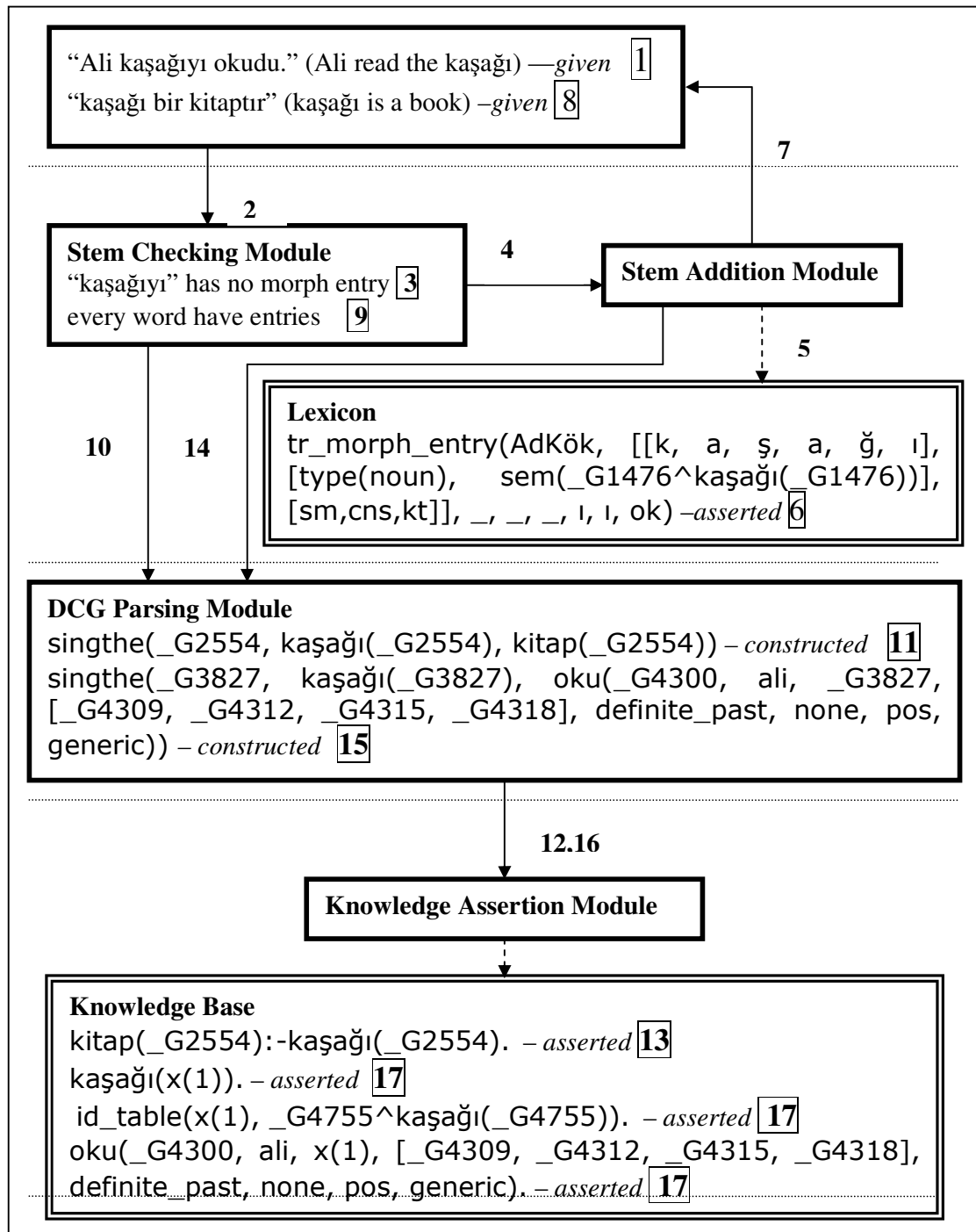


Figure 3.6. Knowledge assertion example

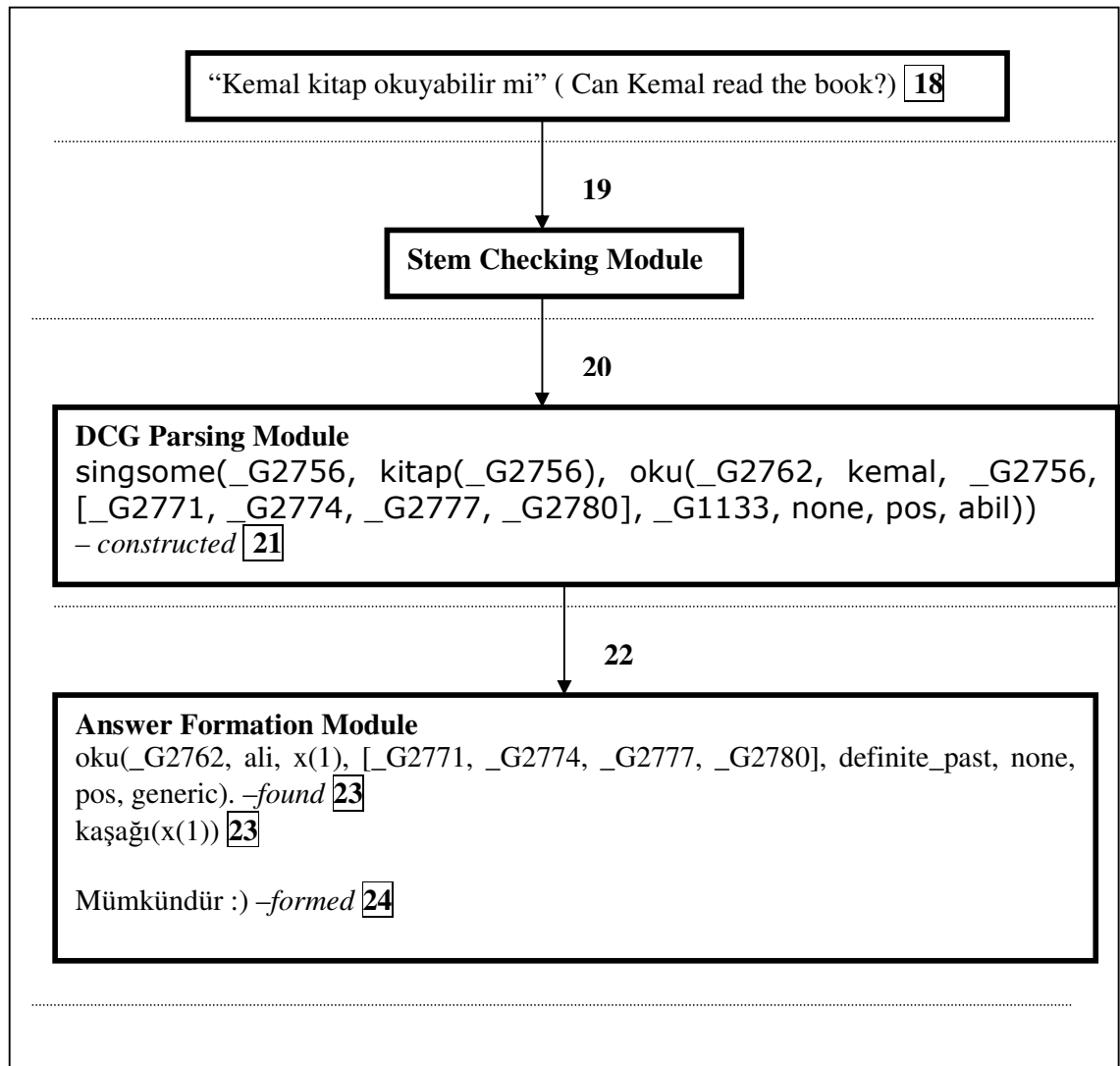


Figure 3.7. Answer formation example

This example illustrates several novel capabilities of TOYagent, like online learning of new word meanings, and a treatment of questions involving deontic modalities. After learning the meaning of “kaşağı”, TOYagent computes the answer to the question by reasoning as follows: Kemal and Ali are both human beings and “kaşağı” is a book. In the sentence “Ali read the kaşağı”, it is mentioned that a human being can read a book. So, as a human being, Kemal can possibly read any book.

4. LEXICON AND MORPHOLOGY

TOY and TOYagent implement the morphological level of NLP by the lexicon and the methods used for generating or parsing the Turkish words composed of the morphemes in the lexicon.

4.1. Lexicon

TOY's lexicon represents a subset of Turkish stems and almost all suffixes using the same data structure. The lexicon contains a small number of sample stems for each syntactic category. If a stem can have more than one surface form due to affixations, a separate entry for each such form is incorporated in the lexicon.

For instance, when a suffix starting with a vowel is added to a word which ends with a voiceless stop, the last sound of the word changes to a voiced consonant. We represent this rule, consonant devoicing, by having an additional lexical entry ending with a voiced consonant for each lexical entry which ends with a voiceless stop.

Kitap alındı.	kitap (normal form)
Kitabı aldı.	kitap + 1 : kitabı (modified form)
	(book) (accusative suffix)

So the “stems” *kitap* and *kitab*, with exactly the same semantics, are both in the lexicon.

In TOY's lexicon, only one lexicon entry exists for each verb. But in TOYagent's lexicon, up to two lexicon entries with different semantic representations can be defined for a verb. The reason for this change is explained in Section 4.2.1.8.

Presently, we have 84 stem entries and 114 derivational and inflectional lexical suffix entries.

4.2. Morpheme Structure

The syntactic categories included in the lexicon are noun, pronoun, verb, adjective, question, time, conjunction, number, and suffixes. A typical lexicon entry is represented by the Prolog structure

```
tr_morph_entry(B,[A,C,D],E,F,G,H,I,J),
```

where

A: The morpheme as a character list

B: Label of the morpheme

C: A list containing an indication of the syntactic category and a lambda expression representing the meaning contribution of the stem or name of the suffix

D: An indication of the location of this stem in the semantic network hierarchy and distinctive features of the stem.

E: Last vowel of the word that the morpheme affixed to

F: Last sound of the word that the morpheme affixed to

G: State of the word that the morpheme affixed to

H: Last vowel of the morpheme

I: Last sound of the morpheme

J: State of the morpheme

Argument A is the surface form of the morpheme represented as a character list.

Argument B shows which syntactic category this morpheme belongs to and can be one of “AdKök” (Nominal root), “ZamirKök” (Pronoun root), “FiilKök” (Verbal root), ”SıfatKök” (Adjective root), “Bağlaç” (Conjunction), “Zaman” (Time), ”Soru” (Question), and “Sayı” (Number). A descriptive string for each different suffix is used in the representation of suffixes. This label is used in the transduction of the string using the Finite State Machines (FSM’s) representing Turkish morphology.

Argument C contains a lambda expression for stems, as will be explained in Section 4.2.1.

Argument D has been added during the development of TOYagent and contains the semantic class and the distinctive features of the morpheme. Semantic classes and distinctive features are explained in Section 4.2.2. These are used in anaphora resolution, which is explained in Section 5.3.3.

In order to create a common data structure for representing both stems and suffixes, this lexicon entry format contains slots which necessarily remain empty for morpheme types for which the features associated with those slots are not relevant.

Arguments E and F are left anonymous in the representation of word stems but they are obligatory for suffix representations. These parameters are used for imposing vowel harmony, and other rules based on the last vowel or letter of the word to which affixation is done. During generation and parsing, these determine which version of the lexical entry for a particular suffix can legally be added to the preceding substring and which can not. Suffix entries are represented as Prolog rules, as in the following example:

```
tr_morph_entry('DH',[[D,H],[tense(definite_past)],[ek(64)],V,L,OK,H,
H,ok):-ltod(L,D),!,
    vtoh(V,H),
    ok(OK).
```

In the representation of the “definite past” suffix, variables D and H are used to cover all the possible surface forms (-dı, -di, -du, -dü, -tı, -ti, -tu, -tü) with only one entry. The correct choice for D is determined by the ltod function, using information about L, the last sound of the preceding word. The last vowel of the preceding word, V, is used by the vtoh function that imposes vowel harmony with H.

Arguments H and I refer to the last sound and last vowel of this morpheme. In stem morphemes, they are not always determined by just taking the last sound and vowel of the stem. In some lexical entries they can differ from the actual values for handling exceptions.

tr_morph_entry('AdKök',[[s,a,a,t],[type(noun),sem(X^hour(X))],[sm,cns,kt]],_,_,_e,t,ok).

Although the last vowel of the word “saat” (hour) is ‘a’, ‘e’ is used in the morph entry, so a derivation such as “saatlar” (*hours) is prevented.

Arguments J and G, representing the “state” of the words with and without this affixed morpheme, respectively, are used in parsing and generation processes to indicate which version of a stem with multiple surface forms started this word, as explained in [1]. “spec” (exceptional forms of special words) and “specok” (standard forms of special words) are two state values used in the lexical entries of the word “çocuk” (the child):

tr_morph_entry('AdKök',[[ç,o,c,u,k],[type(noun),sem(X^child(X))],[sm,cn,i]],_,_,_u,k,specok).

tr_morph_entry('AdKök',[[ç,o,c,u,ğ],[type(noun),sem(X^child(X))],[sm,cn,i]],_,_,_u,ğ,spec).

4.2.1. Semantic Representations of Syntactic Categories in the Lexicon

The semantic representations of morphemes in the lexicon are determined by their syntactic categories. Some categories share the same representation structure.

4.2.1.1. Proper Nouns. Proper nouns are used to uniquely label individuals, and are represented with constants like “Ahmet” in FOPC. Although they are used for identification, personal names are generally used by more than one individual. In TOY, different individuals with the same name are represented with the same constant. So any one of the sentences

“Didem is beautiful, a teacher and a student”,

“There is a beautiful teacher called Didem and a student called Didem”,

“There is a teacher called Didem and there is a beautiful student called Didem”,

where Didem the teacher and Didem the student are different individuals, would cause the same predicates, namely

beautiful('Didem').

teacher('Didem').

student('Didem').

to be added to the knowledge base.

This problem can be solved by enabling the program to use different internal constants for each different individual with the same proper noun.

In TOY's knowledge base, each proper noun is declared as a constant but its semantic representation in the lexicon is of the form

('Didem'^Sco)^Sco,

where Sco is intended to unify with the semantic subformula one of whose arguments will be Didem.

4.2.1.2. Pronouns (Personal and Indefinite). Pronouns refer to other objects like people, places, etc. in a discourse. Different pronouns exist for different types of referent. If a personal pronoun refers to the 3rd person, it must be "o" (he/she/it) and if an indefinite pronoun refers to everybody it must be "herkes" (everybody).

In Turkish, if the subject of a sentence is a personal pronoun, it can be used in the sentence or can be omitted:

Ben çay içtim (I drank tea) : Personal pronoun "ben" (I) is used

Çay içtim (I drank tea) : Personal pronoun is omitted and sentence has no subject.

Although these two sentences have the same meaning, the second one has no subject. In such cases, the personal suffix affixed to the verb determines which personal pronoun is

intended. So in a sentence, the type of the subject pronoun (if it exists) must match the personal suffix used in the verb.

During the processing of a sentence containing a pronoun, the referent must be resolved before proceeding with assertion to the knowledge base. In TOY, no method for anaphora resolution was implemented. As will be explained in Section 5.3.3, TOYagent implements a personal pronoun resolution method in order to process its own target scenarios. Ögün [8] implements resolution of definite phrases with previously created discourse markers and proper nouns. (Discourse markers will be explained in Section 5.4.1.1.)

The semantic representation of a personal pronoun (“sen”-you) in the lexicon is of the form

(singyou^{Sco})^{Sco},

where Sco is intended to unify with the semantic subformula one of whose arguments will be singyou.

The semantic representation of an indefinite pronoun (“herkes”-everyone) in the lexicon is of the form

(X^{Sco})^{plurall(X, person(X), Sco)},

where Sco is intended to unify with the semantic subformula one of whose argument slots will contain the variable X. The type of the quantifier (plurall in this case) in the representation depends on the type of the indefinite pronoun.

4.2.1.3. Common Nouns and Adjectives. As mentioned in Section 3.2, nouns like “peynir” (cheese), “kahvaltı” (breakfast) and adjectives like “küçük” (little), “güzel” (beautiful) are accepted as properties of objects in FOPC and lambda expressions are used to represent such properties.

The semantic representation of a noun (“kahvaltı”-breakfast) in the lexicon is of the form

$X^{\wedge}kahvaltı(X)$.

“Being a dog” is a property of a single object. But “being married” is a property of a pair of objects and the number of arguments that it needs is two. This relationship could be described as $X^{\wedge}Y^{\wedge}evli(X,Y)$. But no such relationship is defined in TOYagent’s current lexicon, and each lambda expression for a noun or adjective has only one argument.

The semantic representation of an adjective (“güzel”-beautiful) in the lexicon is also of the form

$X^{\wedge}güzel(X)$.

4.2.1.4. Conjunctions. Conjunctions are used for combination purposes. In the sentence “Canan elma ve armut yedi” (Canan ate apples and pears) the “ve” (and) conjunction is used between the noun phrases. They can also be used to combine verb phrases or sentences. In TOY, conjunctions were only used to combine noun phrases by using the rule

$$cnp \rightarrow np \text{ conj } np.$$

In TOYagent, the additional $SC \rightarrow S \ S1$ and $S1 \rightarrow CONJ \ SC \mid \varepsilon$ rules are used for parsing conjoined sentences. When conjunctions are used between sentences, the semantic subformulas of the conjunctions themselves are unimportant, as such conjoined sentences receive special treatment. Treatment of conjoined sentences is explained in Section 5.4.2. As a result of this, no modification is done on the representation of conjunctions defined in TOY.

The semantic representation of a conjunction (“ve”/and) in the lexicon is of the form

$Sco^{\wedge}Res2^{\wedge}Res1^{\wedge}X^{\wedge}and(X,Res1,Res2,Sco)$,

where Res1 and Res2 are the restrictors of noun phrases and Sco is intended to unify with the semantic subformula.

4.2.1.5. Question Words. Each question word has its own semantic formula structure according to its meaning. The “kim” (who), “ne” (what), “ne zaman” (when), “hangi” (which), “kaç” (how many- for durations and objects) question words in Turkish have semantic formulas written in terms of the “which” and “how_many” keywords in TOY. Each question word refers to either a person or a place or a time phrase and its representation consists of the referent and one of these keywords as given in Table 4.1. With the exception of kaç (how many), each question word has one entry in the lexicon. “How many” may ask about either objects (“kaç çocuk”-how many children) or durations (“kaç gün”-how many days), so it has one entry for each case in the lexicon.

Table 4.1. Semantic representations of question words

Question Words	Semantic Representation
Kim	$Sco^X^{which(X, person(X), Sco)}$
Ne	$Sco^X^{which(X, object(X), Sco)}$
Ne zaman	$Sco^T^{which(T, time([T, Unit]), Sco)}$
Hangi (which)	$X^{(Res)^{(X^Sco)^{how_many(X, Res, Sco)}}$
Kaç (how many) for duration units	$Sco^{[T, Unit]^{which(T, time([T, Unit]), Sco)}$
Kaç (how many) for objects	$X^{(Res)^{(X^Sco)^{how_many(X, Res, Sco)}}$

These question words are used in the question answering part of TOY, and questions are answered using information stored in knowledge base.

4.2.1.6. Time Units. The time units “dakika” (minute), ”saat” (hour), and “gün” (day) are used as part of time phrases like “5 dakika” (5 minutes) or “9 gün” (9 days).

The semantic representation of a time unit (“dakika”-minute) in the lexicon is of the form

timeunit(dakika).

4.2.1.7. Numbers. The semantic representation of a number (10) in the lexicon is of the form

$X^{\text{number}}(X, [1,0]),$

where the second argument is the digit list of the number.

We do not use question words, time units, and numbers in TOYagent, but their representations still exist in the lexicon.

4.2.1.8. Verbs. The verb in a sentence expresses the described event. In TOYagent, an event has an agent who performs or causes it, a theme which is affected from it, and a location which is the source of origin or the destination of it. Each event must have an agent but may lack a theme or a location. Furthermore, an event may have other constituents like the time of occurrence of this event. The verbs in TOY's lexicon can be divided into two groups: transitive verbs ("içmek"-drink) and intransitive verbs ("uyu"-sleep). A transitive verb is one which has a theme affected by the event, and an intransitive verb is one which does not have a theme. Taking all these into account, transitive and intransitive verbs have different semantic representations:

$\text{Time}^{\text{Loc}}\text{Theme}^{\text{Agent}}\text{verb}(\text{EvNo}, \text{Agent}, \text{Theme}, \text{Loc},$
 $[\text{B}, \text{E}, \text{D}, \text{U}], \text{Tense}, \text{AuxTense}, \text{ST2}, \text{Modality}):$ **for transitive verbs-1**

$\text{Time}^{\text{Theme}}\text{Agent}^{\text{verb}}(\text{EvNo}, \text{Agent}, \text{Theme}, [\text{B}, \text{E}, \text{D}, \text{U}],$
 $\text{Tense}, \text{AuxTense}, \text{ST2}, \text{Modality}):$ **for transitive verbs-2**

$\text{Time}^{\text{Loc}}\text{Agent}^{\text{verb}}(\text{EvNo}, \text{Agent}, \text{Loc}, [\text{B}, \text{E}, \text{D}, \text{U}], \text{Tense},$
 $\text{AuxTense}, \text{Sense}, \text{Modality}):$ **for intransitive verbs**

In TOY, each verb has only one lexical entry, but in TOYagent, in order to prevent anonymous variables in the semantic representation, they can have two entries. Each transitive verb has two lexical entries; one with an 8-ary predicate and one with a 9-ary predicate as shown above. This makes coding of TOYagent easier.

In TOYagent, the name of the verb used in the semantic representation may be different from the actual one like in the lexicon entry of the verb “ol”. This verb has different meanings in Turkish. In the sentence “Canan çocuğu olan bir bayandır” (Canan is a woman who has a child) it means “having something” but in the sentence “Canan mühendis oldu” (Canan became an engineer) it means “to become something”. In TOYagent’s lexicon, to indicate the meaning “having something”, this verb is defined as:

```
tr_morph_entry('FiilKök',[[o,l],[type(verb),sem(Time^Theme^Agent^ var(EvNo,
Agent,Theme,[B,E,D,U],Tense,AuxTense,ST2,Can))],[[agent,sm,cn,i],
[theme,sm,cn,i],[theme,sm,cns,kt]]],_,_,_,o,l,aor).
```

EvNo is a unique number that can be used to distinguish events from each other. This field, which is not used in the current version TOYagent, can play an important role in an application where the temporal order of events is important. Öğün [8] uses EvNo for representing time orderings. If the sentence “Ahmet geldi sonra Mehmet gitti” (Ahmet came, then Mehmet left) is given to his program, the predicates

```
come(evno(1), ahmet, _G358, [_G463, _G466, _G469, _G472], _G352, _G349,
_G346, definite_past, none, pos).
```

```
go(evno(2), mehmet, _G912, [_G1017, _G1020, _G1023, _G1026], _G906,
_G903, _G900, definite_past, none, pos).
```

```
after(evno(1), evno(2)).
```

are asserted to the knowledge base. So the EvNo’s used in the “after” predicate indicate the ordering of the events.

Agent, Theme, and Loc in the representation are the thematic relations or theta roles. Theta roles are the semantic roles assigned to the participants associated with a predicate. The predicate can be either a verb, or an adjective, or a noun. In this subsection, theta roles associated with verbs are discussed. There is no common list of theta roles that is accepted by all linguists and general assignment policy between the participants and the roles.

In [24], six theta roles are defined: agent, theme (patient), experiencer, source, goal and location.

Öğün [8] associated six theta roles with a verb in his work. These are agent (person or thing that causes or actively permits an event), theme (person or thing affected or described), location (where an object is or where an event takes place), goal (result, intended result, or destination), source (point of origin; opposite of goal), and instrument (means with which something is accomplished).

Covington [18] defines seven theta roles: agent, theme, goal, source, benefactive, experiencer and instrument. He proposes that only one participant can be assigned to a theta role, not more. So, he claims that “place” (location) can not be a theta role because in a sentence like “He is living in the black building near the white one at Central Street” there are more than one participant that must be assigned to the location role.

If no phrase is used for a particular theta role in a sentence, this role is left as an anonymous variable in the semantic representation. “Canan çay içti” (Canan drank tea) has no location information, so the location is left as a variable in

singsome(_G551, çay(_G551), _G953^iç(_G956, _G953, canan, _G551, [_G966, _G969, _G972, _G975], definite_past, none, pos, generic)), where _G953 is a free variable and stands for the location.

If more than one phrase corresponds to one theta role in the given sentence, a list consisting of the semantic subformulas of all these phrases is constructed and used in the representation. In the sentence “Canan matematik fizik çalışıyor”, (Canan is studying mathematics and physics), the themes are mathematics and physics. The semantic representation of this event is shown below:

multisome([_G563, _G957], [matematik(_G563), fizik(_G957)], çalış(_G1432, canan, [_G563, _G957], [_G1299, _G1444, _G1447, _G1450], progressive, none, pos, generic))

The list [B, E, D, U] is used for storing the **B**eginning, **E**nd, **D**uration, and (time) **U**nit of an event. This list provides a way of handling adverbial phrases which denote time. If no time phrase is used in the given sentence, these arguments are left free. The only time unit implemented in TOY is “saat” (hour), and if two of the B, E, and D arguments are bound, the remaining free one can be evaluated using arithmetic operations using modulo 24, i.e. $End = (Beginning + Duration) \bmod 24$.

The value of “Tense” is derived from the time suffix affixed to the verb of the sentence. Possible tense values are *definite_past* (“uyudu”-slept), *narrative_past* (“uyumuş”- has apparently slept), *progressive* (“uyuyor”-is sleeping), *future* (“uyuyacak”- will sleep), *aorist* (“uyur”-sleeps), *subjunctive* (“uyusa”- were to sleep), *necessitative* (“uyumalı”-must sleep), and *imperative* (“uyu”- sleep). All of these tense names except aorist are used in the representation. If the tense of a verb is aorist, the tense argument is left as an anonymous variable, so that the associated knowledge item is true regardless of time.

A verb may or may not have an auxiliary tense. Each auxiliary tense suffix comes after the tense suffix in a verb. Possible auxiliary tense values are *definite_past* (“uyuyordu”-was sleeping), *conditional* (“uyuyorsa”-if (he) is sleeping), *dubitative* (“uyuyormuş”-it is said that (he) was sleeping), or *none* (“uyuyor”-is sleeping).

The sense of the verb can be either affirmative or negative. If the verb has a suffix that imposes negative meaning, the sense is negative, otherwise, it is affirmative.

Canan uyu+ **ma**+ dı (Canan didn't sleep): sense is negative
 root+ negative suffix + tense suffix

Canan uyudu (Canan slept): sense is affirmative

In TOY, the sentences “Canan kahvaltıda çay içer” (Canan drinks tea at breakfast) and “Canan kahvaltıda çay içebilir” (Canan can drink tea at breakfast) are treated identically. In the second sentence, it is said that Canan can drink tea at breakfast; she has the capability of drinking tea, but no information about whether she drinks tea or not is

given. In the first sentence, it is said that she generally drinks tea at breakfast. In order to make this distinction, a Deontic Modality argument has been added to the semantic representation of verbs in TOYagent. This argument has the possible values “generic” or “abil”.

“Canan uyur” (Canan sleeps)

uyu(_G1199, canan, _G1120, [_G1117, _G1211, _G1214, _G1217], aorist, none, pos, generic)

“Canan uyuyabilir” (Canan can sleep)

uyu(_G1373, canan, _G1240, [_G1237, _G1385, _G1388, _G1391], aorist, none, pos, abil)

This deontic modality feature improves the question answering capability of TOYagent and is further explained in Section 6.2.

4.2.1.9. Suffixes. Instead of a semantic formula like those of the other categories, only a list including an indication of the meaning of the suffix is used. This list shows the features that will be associated with the formula of the word to which this suffix is affixed:

[tense(definite_past)] is the feature list defined for the definite past suffix (-dH)

The suffixes affixed to the word are combined into a list during its parsing. They provide a great deal of information about this word like number, tense, case, etc.

4.2.2. Semantic Classes and Distinctive Features

The semantic class that the morpheme belongs to and its distinctive features, which will be introduced shortly, are stored in its lexicon entry (Part D). The semantic class that the word belongs to is represented by a path in the semantic network hierarchy as explained in Section 3.4. The list inserted to the slot (Part D) has two parts which are divided by “[]”.

The representation of the morpheme peynir (cheese):

```
tr_morph_entry('AdKök',[[p,e,y,n,i,r],[type(noun),sem(X^peynir(X))],[sm,cns,kt,
[],yn,krm]],_,_,_i,r,ok).
```

[sm,cns,kt,[],yn,krm] is the inserted list, [sm,cns,kt] is the semantic class and [yn,krm] represents the distinctive features.

As seen in the above entry, abbreviations in Turkish, which are hand coded, are used to represent the classes and the features in the semantic network hierarchy given in Figure 3.3. The semantic class abbreviations are shown in Table 4.2 and some of the distinctive feature abbreviations are shown in Table 4.3.

Table 4.2. Semantic class abbreviations

Semantic classes	Abbreviations
Concrete	sm
Abstract	Sy
Animate	cn
Inanimate	cns
Human being	i
Animal	hy
Plant	bt (Bitki)
Solid	Kt (Katı)
Liquid	Sv (Sıvı)
Gas	hv (Hava)

Table 4.3. Distinctive features and abbreviations

Feature	Abbreviation
“bayan” (woman)	byn
“bay” (man)	by
“polis” (police)	pols
“yenebilir” (eatable)	yn
“kırılmaz”(unbreakable)	krm
“içilebilir” (drinkable)	il
.....

The semantic network we constructed in TOYagent is not sufficiently detailed for handling all the variety among entities described by words. Obtaining a general classification hierarchy that covers all entities in the external world requires great amount of time and effort like the Cyc ontology, which has been under construction by a large number of developers over the past dozen years [11]. The aim of this thesis is not the construction of a complete semantic network hierarchy for Turkish words, but just a “proof of concept” therefore we defined a few classes that can form a basis for an overall network.

As mentioned in Section 2.2, the arcs in a semantic network can represent different kind of relations (subset, likes, hates, etc). The only relation type that we implemented in our network is the “subset” relation as mentioned in Section 3.4. Coupled with the fact that the number of classes in our hierarchy is very small, this results in lots of entities that have many different features from each other belonging in the same class. The “liquid” class includes “çay” (tea), “kahve” (coffee), and “alkol” (alcohol) but these have different features, for instance, two of them (tea and coffee) are hot beverages, but alcohol is not a hot beverage.

The features that we keep in the distinctive features list can be divided into two groups: the ones that indicate subsets of this semantic class, and that would therefore become unnecessary if we had a bigger network with more classes, and the ones that indicate other relations like “gender-of” or “job-of”, which would have to stay in the data structure even for a bigger hierarchy, demonstrating the need for the “distinctive features” sublist.

The representation of the morphemes Kemal and Canan look like:

```
tr_morph_entry('AdKök',[['K',e,m,a,l],[type(propernoun),
sem((kemal^Sco)^Sco)],[sm,cn,i,[],by,pols]],_,_,_,e,l,ok).
```

```
tr_morph_entry('AdKök',[['C',a,n,a,n],[type(propernoun),
sem((canan^Sco)^Sco)],[sm,cn,i,[],byn,pols]],_,_,_,e,l,ok).
```

In this representation the feature “by” and “byn” are subset features and the “pols” is a job-of feature.

If a detailed classification of all entities is implemented in our hierarchy, the subset relations of all morpheme entries in TOYagent’s lexicon will become classes of our network. These classes (nodes) become subsets of the classes at the left side of “[]” character in their corresponding morpheme entries, “i” (human being) in this case. In this case our network will look like the one shown in Figure 4.1, if it is detailed.

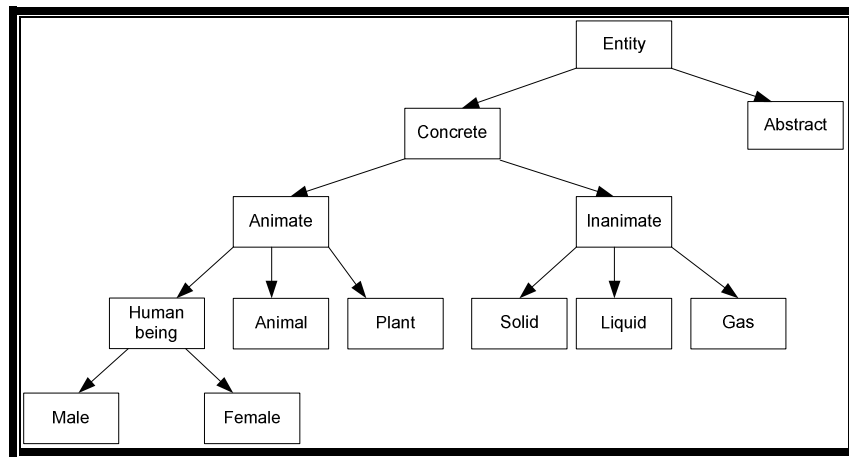


Figure 4.1. Improved semantic network hierarchy

After this classification, Canan and Kemal belong to different classes but they share the common feature of being in the police force. In our network, each arc represents only the subset relation, and as a result, this kind of features will always be linked to the entities as a feature list until this restriction is removed from the system, by the introduction of different arc types such as “job-of”.

In the current system, a morpheme may lack distinctive feature list in its morpheme entry. The morpheme “sınıf” (classroom) has no such list:

```
tr_morph_entry('AdKök',[[s,i,n,i,f],[type(noun),sem(X^sınıf(X))],[sm,cns,kt]]
,_,_,_,i,f,ok).
```

As described in Section 4.2., the same data structure is used for each morpheme. The slot reserved for storing the semantic class and features has the following different uses for morphemes with different syntactic categories:

- For personal/indefinite pronouns, time units, question words, and conjunctions, an empty list is inserted into this slot:

```
tr_morph_entry('ZamirKök',[[b,e,n],[type(personalpronoun),sem((i^Sco)^Sco),
per(1),num(sing)],[ ]],_,_,_,e,n,ok).
```

- In noun morphemes, the list explained above is inserted:

```
tr_morph_entry('AdKök',[['C',a,n,a,n],[type(propernoun),sem((canan^Sco)^
Sco)],[sm,cn,i,[ ],byn]],_,_,_,a,n,ok).
```

(Canan is a human being and a woman.)

```
tr_morph_entry('AdKök',[[e,t], [type(noun), sem(X^et(X))], [sm,cns,kt]] ,_, _ , _ ,
e,t,ok).
```

(Meat is a solid object.)

```
tr_morph_entry('AdKök',[[i,ç,e,c,e,k], [type(noun),sem(X^içecek(X))],
[sm,cns,sv,[ ],il]],_,_,_,e,k,ok).
```

(Beverage is a liquid object.)

- In adjective morphemes, a list of the semantic classes and the features of the nouns which can reasonably be modified by this adjective is inserted:

```
tr_morph_entry('SifatKök',[[l,e,z,z,e,t,l,i],[type(adjective),sem(X^lezzetli(X))],
[sm,cns,sv],[sm,cns,kt]]],_,_,_,i,i,ok).
```

(A noun like “et” (meat) which is a solid object or a noun like “içecek” (beverage) which is a liquid object can be modified by this adjective. So, “lezzetli et “ (delicious meat) can be a possible combination.)

- In verb morphemes, the list of the semantic classes and the features of possible agents, themes and locations of the sentences where this morpheme can be used as the verb is inserted:

```
tr_morph_entry('FiilKök',[[i,ç],[type(verb),sem(Time^Theme^Agent^
iç(EvNo,Agent,Theme,[B,E,D,U],Tense,AuxTense,ST2,Can))],
[[theme,sm,cns,sv,[ ],il],[agent,sm,cn,i,[ ],byn], [agent,sm,cn,i,[ ],by],
[loc,sm,cns,kt]]],_,-,_,i,ç,aor).
```

(The agent of this verb must be a human being (either male or female), the theme must be a liquid, and the location must be a solid inanimate object like the table.)

- In suffix morphemes, the suffix number is inserted into this slot:

```
tr_morph_entry('singnoun',[[],[num(sing)],[ek(6)]],V,L,OK,V,L,OK)
```

This number is unique for all suffixes and it is used by the sentence generation mechanism.

4.3. Generation and Parsing

Finite state machines, based on Oflazer's [25], are used in parsing and generation processes. The first FSM is used for words which have nominal roots. The second FSM is designed for verbal roots. The nominal FSM is given in Figure 4.2 and the verbal FSM is given in Figure 4.3 and 4.4.

The arguments in the lexical entries and the morphemes themselves are used while traversing these FSMs. In TOY, each FSM is implemented by defining the initial node, the final nodes and the arcs used in it. The initial node and one of the final nodes and one of the arcs is shown in Table 4.4.

Table 4.4. Definitions of nodes and arcs in FSMs

initial(1)	initial node
final(301)	one of the final nodes
arc(113,123,'DH')	one of the arcs used

A suffix with the label 'DH' (see the explanation in Section 4.2) causes the parser to jump to node 123 from node 113 in the FSM. These labels provide unique identification for suffixes, whereas all stems of the same syntactic category share the same label.

In TOY, a nominal verb can be used in a statement sentence and this sentence can be parsed using the verb phrase \rightarrow noun phrase rule.

“Canan çocuktur” (Canan is a child) can be processed and accepted by TOY.

But if a nominal verb is used in a yes/no question, like “Canan çocuk mudur?” (Is Canan a child?), this sentence can not be parsed. In yes/no questions, the verb must have the 'mH' suffix, which is a question tag. Other suffixes may be affixed to this suffix, as seen in the sentence “Canan çocuk mudur?”.

A nominal verb given to TOY's parser ends with a final node by traversing one of the nodes labeled as “nominal verb nodes” (nodes 14, 17, 21 in Figure 4.2). No arc labeled 'mH' is defined as going from these nodes to others, so that no nominal verb can have the yes/no question suffix affixed to itself.

In order to solve this problem, new arcs have been added to the FSM's in TOYagent. The parsing operation of a nominal verb in a statement sentence is again done using the nominal FSM. But the parsing of nominal verbs in yes/no questions is transferred to the verbal FSM. New arcs are added to the node labeled as “nominal root” (Node 2). They point to some of the nodes in verbal FSM, which have arcs with label “-mH” or “simple_mH”. This method is applicable because any suffix that can be affixed to the root of a nominal verb can also be affixed to a verbal root. Also the arcs going out from these nodes in the verbal FSM cover all possible suffixes that may be affixed to a nominal verb

after the 'mH' suffix. So, the set of all possible suffixes that may come after or before the 'mH' suffix in a nominal verb are covered in this method. The added arcs and the affected nodes are highlighted in Figure 4.4.

The root matching (left-to-right) parsing method is used in the morphological stage. In this method, starting from a prefix of the given word, the parser tries to find the possible affixed suffixes until a final node and end of the word are reached. The affixed suffixes are found by comparing the suffix morphemes in the lexicon and possible substrings of the remaining word after the stem. Each affixed suffix makes the parser jump from a node to another in the FSMs. If the parser proves the legality of the word, the prefix token at the beginning becomes the stem of the given word. Outputs of this parsing process are two output lists, one of them includes all the C arguments and the other includes all the D arguments of the morphemes used during the parsing process.

- The structure of the first list is
[type of the word, semantic formula of the word, names of the affixed suffixes]
- The structure of the second output list is
[the semantic class of the stem, suffix numbers of the affixed suffixes]

The second output list has been added in TOYagent and is used by the anaphora resolution routine, which will be explained in Section 5.3.3.

When a word of the input sentence is parsed using the “parse” predicate, an entry for that word is added to the “temp_words_content” list. This entry has the structure:

[word, first output list retracted from its parse].

In TOYagent, if the word is given as input, the output lists are retrieved as given in Figure 4.5. By using the reversibility advantage of Prolog programming, if the first output list is given as input and the second one is left anonymous, the possible words that can have this list are retrieved, as given in Figure 4.6.

```
?-parse([i,ç,t,i],A_list1,A_list2).

A_list1 = [type(verb), sem(_G324^_G327^_G330^iç(_G333, _G330, _G327,
[_G342, _G345, _G348, _G351], _G337, _G338, _G339, _G340)), _G286, _G287,
_G288, _G289)), st2(pos), tense(definite_past), per(3), num(_G433),
st1(statement), aux tense(none)]

A_list2 = [[[theme, sm, cns, sv, [], il], [agent, sm, cn, i]], [ek(115)], [ek(64)], [ek(46)],
[ek(113)]] .
```

Figure 4.5. Parsing a word

```
?-parse(Word, [type(verb), sem(_G324^_G327^_G330^iç(_G333, _G330, _G327,
[_G342, _G345, _G348, _G351], _G337, _G338, _G339, _G340)), _G286, _G287,
_G288, _G289)), st2(pos), tense(definite_past), per(3), num(_G433),
st1(statement), aux tense(none)],_).

Word= [i, ç, t, i];
Word = [i, ç, t, i, l, e, r].
```

Figure 4.6. Generating a word

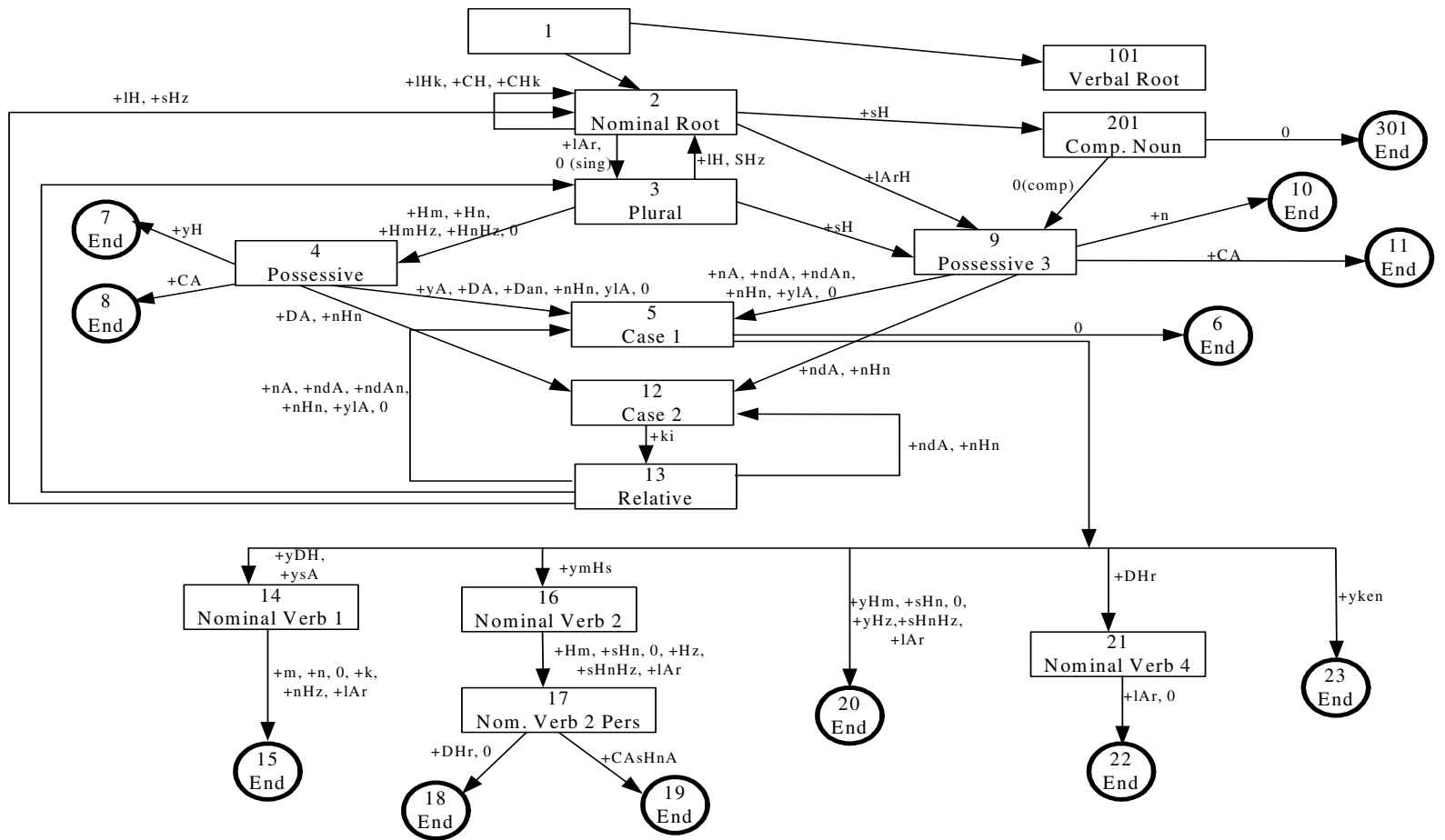


Figure 4.2. Finite State Machine for Nouns

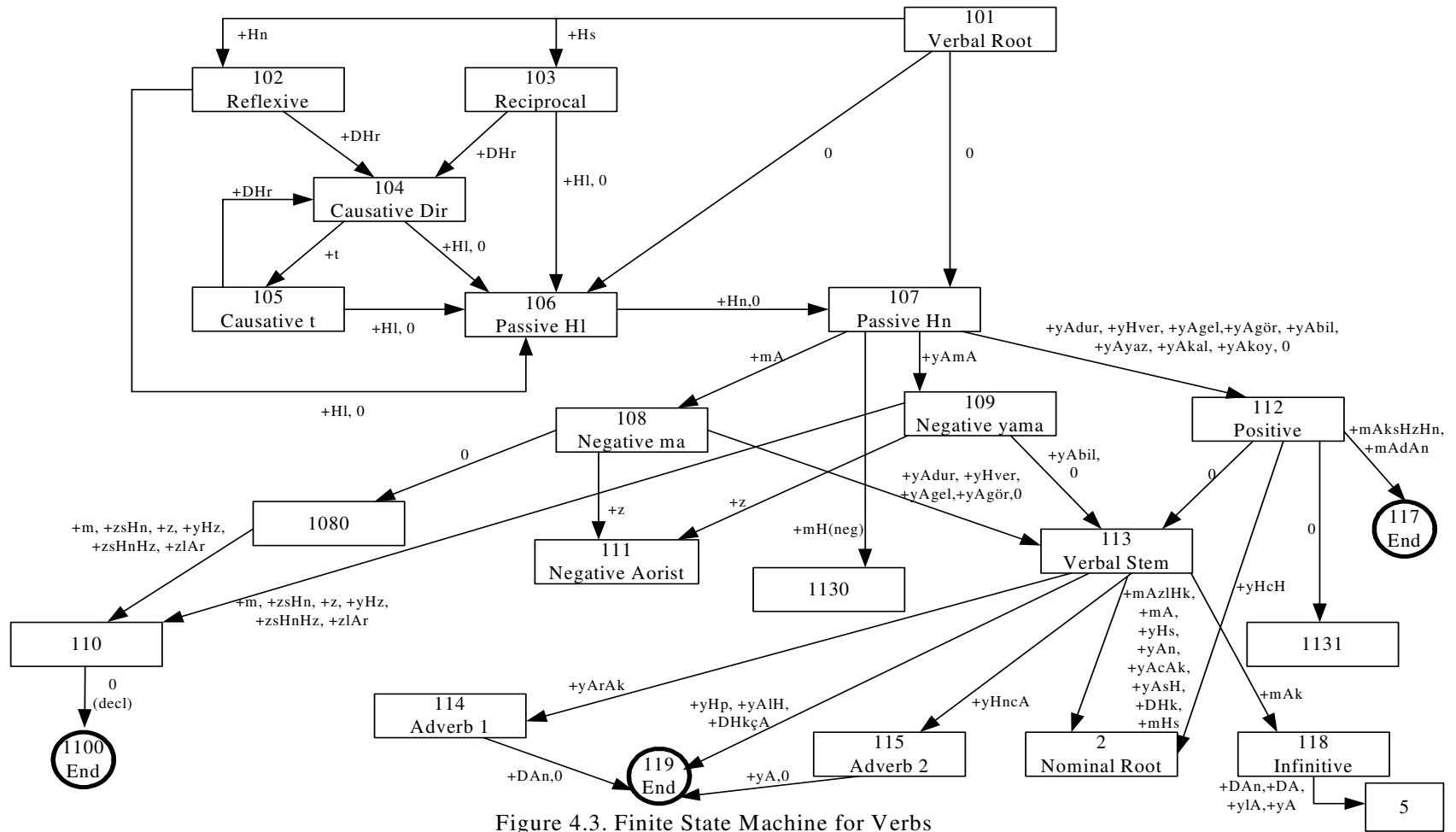


Figure 4.3. Finite State Machine for Verbs

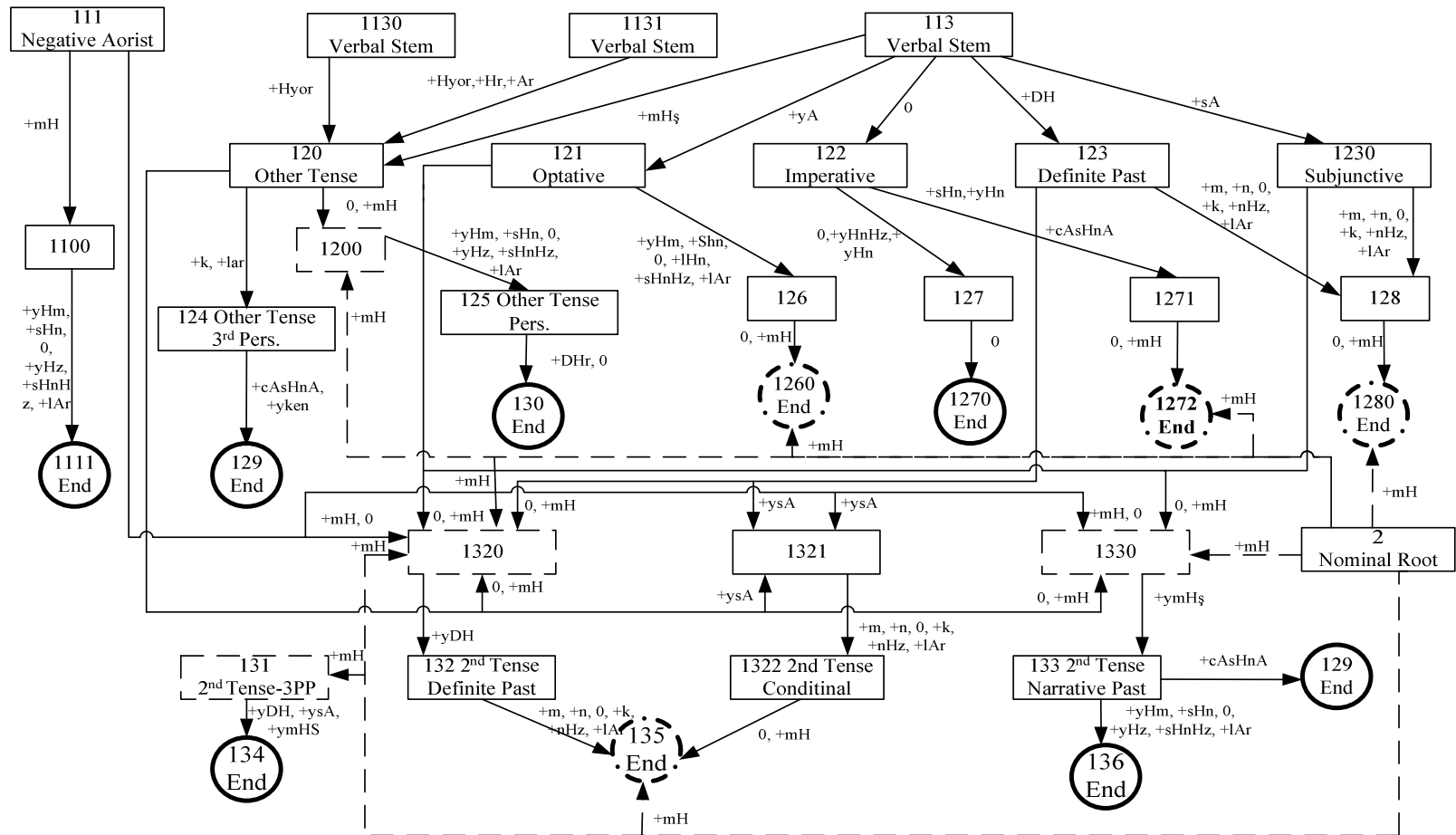


Figure 4.4. Finite State Machine for Verbs (continued)

“Koyun” is a word which has more than one meaning, one of which is the noun koyun (sheep) and another is the verb koyun (put it). The correct meaning of this word is determined according to its usage in the sentence:

“Koyun meliyor” (The sheep is bleating)

“Çaydanlığı masaya koyun” (Put the teapot on the table)

If such an ambiguous word is given to the TOYagent’s parser, it can find all possible stems, as shown in Figure 4.7.

```
?- parse([k,o,y,u,n],A_list1,A_list2).
A_list1 = [type(verb), sem(_G333^_G336^_G339^koy(_G342, _G339, _G336,
[_G351, _G354, _G357, _G360], _G346, _G347, _G348, _G349)), _G298, _G299,
_G300, _G301)), st2(pos), tense(imperative), per(2), num(plur), st1(imperative),
auxtense(none)]
A_list2 = [[], [ek(109)], [ek(115)], [ek(82)], [ek(56)], [ek(114)]] ;

A_list1 = [type(verb), sem(_G333^_G336^_G339^koy(_G342, _G339, _G336,
[_G351, _G354, _G357, _G360], _G346, _G347, _G348, _G349)), _G295, _G296,
_G297, _G298)), st2(pos), tense(imperative), per(2), num(plur), st1(imperative),
auxtense(none)]
A_list2 = [[], [ek(115)], [ek(82)], [ek(59)], [ek(114)]] ;

A_list1 = [type(noun), sem(_G291^koyun(_G291)), num(sing)]
A_list2 = [[sm, cns, hy], [ek(6)]].
```

Figure 4.7. Parsing an ambiguous word

If the word “koyun” in the sentence “Koyun meliyor” (The sheep is bleating) is sent to the parser, the initial stem that is found by the parser is not the correct one for this interpretation. Having no access to the sentence context, the parser may not find the correct stem of a word at the first turn. However, the correct one will be found in one of the following turns.

5. ACQUIRING KNOWLEDGE FROM THE USER

TOYagent is a communication tool between the user and the computer. The two main operations performed by this program are obtaining knowledge and answering questions about the knowledge it has. The operation performed is based on the user input. If the mood of the sentence that the user entered is 'statement' then the agent tries to extract the knowledge contained in it and assert that piece of knowledge in its knowledge base. But if the mood is 'yes/no question', the agent tries to form an answer to this query based on the present knowledge. User inputs and program outputs are expressed by Turkish natural language sentences. Knowledge accumulation is explained in this chapter and question answering will be explained in Chapter 6.

If the mood of the input sentence is 'statement', initially its constituents are morphologically checked. Then, the structure of the sentence is syntactically checked and finally, the embedded knowledge is asserted after semantic processing. TOYagent continues its execution until the user enters "hoşçakal" (good bye). Words in a sentence must be separated with blanks, and a period must be placed at the end of the input sentence as seen in Figure 5.1.

<p>Türkçe Okur Yazar (Literate in Turkish) Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)</p> <p>Cümleniz> çocuğu olan bayan annedir. (A woman who has a child is a mother) Cümleniz> hoşçakal. (good bye)</p>
--

Figure 5.1. TOYagent interface

Initially, the sentence is sent to the 'lexical preprocessor' module, where all its words are morphologically checked. Then, the DCG parsing of the sentence is done and its semantic formula is constructed if possible. Finally, using the semantic formula, the information in the sentence is added to the knowledge base.

5.1. Lexical Preprocessor

In TOY, during the syntactical processing of the given natural language sentence, constituents of the sentence are sent to the morphological parser. When a word is successfully transduced through the FSM's, it is proved that this word is legal for Turkish. But this application is based on an assumption that the stem of each constituent word has an entry in the lexicon with the structure "tr_morph_entry". As a result of this assumption, a sentence containing one or more words which have no entries in the lexicon causes TOY to terminate by outputting "No" on the screen. No control is performed on the given sentence before processing it, and this causes essentially morphological errors to be treated as syntactical errors by the program. This assumption also makes TOY a program that has a processing range which is determined by the number of words in the lexicon.

In TOYagent, we add stem checking and stem addition modules to TOY in order to remove these control and range problems. The stem checking module parses each word in the given natural language sentence before sending it to the "get_sem" predicate. If this module finds words that have no entries in the lexicon, then these are sent to the stem addition module. The stem addition module adds new entries to the lexicon for these words. Some parts of these entries, like the last vowel of the word, can be determined by the program automatically; some, like the correct stem of the word, need user specifications. As a result of this, user interaction is needed in this module. These modules broaden the processing range of the program by providing the capability of adding new word entries to the lexicon during the execution. They also provide a way for the program to learn the descriptions of these new words. If the stem addition module is executed for any word, then TOYagent asks the user to define the word by giving its description(s).

The entries in the lexicon are divided into ten groups in TOY as explained in Section 4.2. These are proper nouns, indefinite pronouns, personal pronouns, nouns, adjectives, verbs, time_units, questions, conjunctions and suffixes. All personal pronouns and suffixes in Turkish are already in TOY's lexicon. Only words of type proper noun, noun, adjective or time_unit can be added to the lexicon in the stem addition module. We have reasons for not supporting such "on-line" learning of words in other categories: Verbs are either transitive or intransitive, and the number of parameters used in the morph entry differs

according to this distinction, as explained in Section 4.2.1.8. In order to add a correct entry for a verb, the program has to ask “Is this verb transitive or intransitive?” to the user. Such questions presume a much higher level of linguistic knowledge on the part of the user than is acceptable to us in this project. Semantic representations of questions differ according to whether their referents are people, places, time phrases, etc., as explained in Section 4.2.1.5. Again, a question that requires linguistic knowledge, as explained before, has to be asked to the user for adding a correct question entry to the lexicon. For each conjunction entry, a transformation rule is implemented in TOYagent as explained in Section 5.4.2. Learning of this rule from the user can not be performed easily by asking questions.

5.1.1. Stem Checking Module

This module is responsible for taking each word of the sentence, except the determiners, sequentially and checking whether there is an entry for this word in the lexicon or not. The natural language sentence that will be checked is given to the “tscontrol(Sentence, List)” predicate. The Sentence argument is the input sentence in word list format. The List argument is initially a free variable but after the predicate is executed, it includes the unrecognized words in Sentence if any exist. This list is given to the stem addition module for addition operation.

Cümleniz> **küçük çocuk leoparı görmüş. (The little child has apparently seen the leopard)**

This sentence is given to the tscontrol predicate: tscontrol([küçük,çocuk,leoparı,görmüş],List). “küçük” (little), “çocuk” (child) and “gör-” (to see) have entries in the lexicon. After this predicate is executed, List includes the only unrecognized word “leoparı” (leopard + accusative), and it is sent to the stem addition module. The agent outputs “leoparı kelimesini bilmiyorum” (I don’t know the meaning of the word “leoparı”).

5.1.2. Stem Addition Module

This module is responsible for learning the words identified by the stem checking module from the user. Any mistake in the declaration of the morpheme structure affects the execution of TOY's morphological parser. Some arguments of the morpheme structure can be determined by the agent automatically, but some of them can not. So, TOYagent asks questions to the user when it needs help for the determination of an argument.

Insertion of the word to TOYagent starts with the construction of its morpheme entry and ends with the insertion of its description to the lexicon. A verbal description of each unknown word must be given by the user after its entry is inserted to the lexicon.

The additions done in this module can be used only in that conversation session, not in the following ones.

5.1.2.1. Construction and Insertion of the Morpheme Entry. Formally, each lexical entry has the structure `tr_morph_entry(B,[A,C,D],E,F,G,H,I,J)`, as explained in Section 4.2. Required arguments for morpheme entries of each syntactic category are different and their determinations are done either automatically or by the user.

Initially, the morpheme entry has the value:

```
tr_morph_entry(,[_,_,[]],_,_,_,_,_).
```

Argument A: If in the sentence “küçük çocuk leoparı görmüş” (The little child has apparently seen the leopard), ‘leoparı’ (the leopard) is determined as an unknown word by the stem checking module, the root of this word (“leopar” (the leopard)) must be inserted to this slot in the structure. So, the root of the unknown word must be found before it is inserted.

A stemming operation is performed in order to obtain the root(s) of a word and the “`parses(CList)`” predicate is used for this purpose, where CList is the unrecognized word.

In order to take consonant devoicing into account, if a root found by the “parse” predicate ends with b, c, d, or ğ, one more choice is offered to the user in which the last sound is changed with p, ç, t, or k, and vice versa:

Cümleniz> **mektebe geldim. (I came to the school)**
 T_agent: mektebe kelimesini bilmiyorum (I don't know the meaning of the word “mektebe”)
 Kelimenin Kökü hangisi: (Which one is the root of the word?)
 1: mekteb
 2: mektebe
 3: mektep
 Hangisi (Which one): **3.**

As seen in the example, if this case is not handled, the correct root of the word “mektebe” (to the school) can not be found.

Argument B: The syntactic category of the unknown word can not be determined easily by the agent, so it is asked to the user. The possible categories are displayed by the agent, and user selects which category this word belongs to:

T_agent: leopar kelimesinin türü nedir? (What is the type of the word “leopar”)
 1:İsim (Noun)
 2:Sıfat (Adjective)
 3:Zaman(Time_unit)
 4:Özel İsim (Propernoun)
 Hangisi (Which one):**1.**

If the user selects one of them, the label is inserted to the structure:

tr_morph_entry('AdKök',[[l,e,o,p,a,r],[_,_],[[]],_,_,_,_,_]).

Argument C: The construction of this part is done automatically by using the user selections about the previous parts. The semantic representations of each syntactic category were explained in Section 4.2.1. Using the category chosen by the user, the correct lambda expression is determined.

The root of the word and the lambda expression are combined and inserted to the structure:

```
tr_morph_entry('AdKök',[[l,e,o,p,a,r],[type(noun),sem(X^leopar(X))],[[]],_,_,_,_,
_,_).
```

Argument D: The list that indicates the semantic class is constructed according to responses received to questions asked to the user. The list that shows the word's distinctive features is left blank and it is combined with the semantic class. Finally, this combined list is inserted to the structure.

The program asks the user whether he/she wants to give semantic information about the word or not. If the user does not want to do this, this question part is skipped and an empty list is inserted. Words with such empty lists can not be used during personal pronoun resolution.

<p>T_agent: "leopar kelimesinin tanımıyla ilgili sorular sorayım mı?(e/h) !: e. (yes) T_agent:Aşağıdaki sınıflardan hangisine giriyor: somut soyut > somut. (concrete) T_agent:Aşağıdaki sınıflardan hangisine giriyor: canlı cansız > canlı. (animate) T_agent:Aşağıdaki sınıflardan hangisine giriyor: insan hayvan bitki > hayvan. (animal)</p>

The combined list is inserted to the structure:

```
tr_morph_entry('AdKök',[[l,e,o,p,a,r],[type(noun),sem(X^leopar(X))],
[sm,cn,hy]],_,_,_,_,_).
```

A shortcoming of the stem addition module is the fact that distinctive features are not acquired about the newly learned words. A better mechanism must be added to the agent for this purpose in the future.

Arguments E, F, and G: These arguments are left blank in the morpheme entry of a word if its syntactic category is not “suffix”. Since new suffixes can not be added to the lexicon, in the structure these arguments are left blank.

Arguments H, I, J: The values inserted to these arguments depend on the last sound and the last vowel of the word. As a result of this, the program initially finds the last sound and the last vowel of the word. If the last sound is a vowel, then these values are inserted to arguments H and I and “ok” is inserted to argument J.

But the determination of these arguments is not so simple if the last sound is a consonant. Two important cases have to be taken into account during the determination process: consonant devoicing and vowel insertion.

In Turkish, consonant devoicing may occur in some cases as explained in Section 4.1, so that, if the last sound of the word is a voiceless stop, we add two entries for this word. In the first one, the last sound and the last vowel found before are inserted to arguments H and I and the state is set to “specok”. In the second one, the state is set to “spec” and the last sound of the word after consonant devoicing is inserted to argument H. Argument I remains the same.

```
tr_morph_entry('AdKök',[[e,k,m,e,k],[type(noun),sem(X^ekmek(X))],
[sm,cns,kt]],_,_,_,e,k,specok).
```

```
tr_morph_entry('AdKök',[[e,k,m,e,ğ],[type(noun),sem(X^ekmek(X))],
[sm,cns,kt]],_,_,_,e,ğ,spec).
```

All of the insertions defined above are done automatically by the program, but user interaction is needed for the vowel insertion case. In Turkish, when a suffix starting with a vowel is affixed to a word ending with a consonant, the last vowel of the word may be dropped. The possibility of this occurrence depends on the word:

When the –H suffix is added to the word “kayın” (brother-in-law), it turns to “kayını” (his/her brother-in-law). But when the same suffix is sent to the other sense of “kayın” (beech), the result is “kayını”.

In order to take vowel insertion into account, two entries are added to TOY’s lexicon. In the first one, the word itself is used but in the second entry the word that remains after dropping the vowel is used. The state is set to “specok” in the first entry and “spec” in the second one. The last sound and the last vowel of the word used in each entry are bound to arguments H and I.

Since the occurrence of vowel insertion depends on the meaning of the word, the program drops the last vowel from the word and asks the user whether this word is legal or not. According to the user response, the second entry is created:

T_agent: Bu kelimenin kimi çekimlerinde ön kısmı [k,a,y,n] şeklinde yazılır mı?
e(evet)/h(hayır) (Does the prefix “kayn” appear in some inflections of this word?)
User: e. (yes)

```
tr_morph_entry('AdKök',[[k,a,y,i,n],[type(noun),sem(X^kayın(X))],[sm,cns,kt],
_,_,_,i,n,specok).
tr_morph_entry('AdKök',[[k,a,y,n],[type(noun),sem(X^kayın(X))],[sm,cns,kt]],
_,_,_,a,n,spec).
```

If consonant devoicing or vowel insertion are not applicable on the unknown word, its last vowel and last sound are used in the entry and the state is set to “ok”.

T_agent: Bu kelimenin kimi çekimlerinde ön kısmı [l,e,o,p,r] şeklinde yazılır mı?
 e(evet)/h(hayır) (Does the prefix “leopr” appear in some inflections of this word?)
 User: **h. (no)**

```
tr_morph_entry('AdKök',[[l,e,o,p,a,r],[type(noun),sem(X^leopar(X))],
[sm,cn,hy]],_ ,_ ,_ ,a,r,ok).
```

5.1.3. Obtaining a Description of the Unknown Word

After a morph entry is added to the lexicon for the unknown word, TOYagent asks the user to describe the word in terms of other words. The user can give one or more descriptive sentences at the same time. These new sentences are again processed in the way described in this chapter. Since the same operations are applied, unknown words in these descriptive sentences are also added to the lexicon during the execution as shown in Figure 5.2, 5.3, 5.4 and 5.5.

Türkçe Okur Yazar (Literate in Turkish)
 Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)
 Cümleniz> **küçük çocuk leoparı görmüş. (The little child has apparently seen the leopard)**
 T_agent: leoparı kelimesini bilmiyorum (I don't know the meaning of the word “leoparı”)
 Kelimenin Kökü hangisi: (Which one is the root of the word?)
 1 : leopar
 2 : leoparı
 Hangisi(Which one): **1.**
 T_agent: leopar kelimesinin türü nedir ? (What is the type of the word “leopar”)
 1:İsim (Noun)
 2:Sıfat (Adjective)
 3:Zaman (Time_unit)
 4:Özel İsim (Propernoun)
 Hangisi(Which one):**1.**

Figure 5.2. Sample dialogue

T_agent: “leopar kelimesinin tanımıyla ilgili sorular sorayım mı?(e/h)
 l: **e.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:
 somut soyut
 > **somut.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:
 canlı cansız
 > **canlı.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:
 insan hayvan bitki
 > **hayvan.**

T_agent: Bu kelimenin kimi çekimlerinde ön kısmı [l,e,o,p,r] şeklinde yazılır mı?
 e(evet)/h(hayır) (Does the prefix “leopr” appear in some inflections of this word?) **h.**

tr_morph_entry(AdKök, [[l, e, o, p, a, r], [type(noun),
 sem(_G1557^leopar(_G1557))], [sm,cn,hy]], _, _, _, a, r, ok)

T_agent:leopar ne demek? Anlatır mısın? (What does “leopar” means? Can you explain?)
 l: **bütün leoparlar vahşidirler ve leopar etle beslenir. (All leopards are wild and leopards eat meat)**

T_agent:etle kelimesini bilmiyorum (I don’t know the meaning of the word “etle”)
 Kelimenin Kökü hangisi: (Which one is the root of the word?)
 1 : et
 2 : etl
 3 : etle
 4 : ed
 Hangisi (Which one): **1.**

Figure 5.3. Sample dialogue (continued)

T_agent:et kelimesinin türü nedir ? (What is the type of the word “et”)

1:İsim (Noun)
 2:Sıfat (Adjective)
 3:Zaman(Time_unit)
 4:Özel İsim (Propernoun)
 Hangisi (Which one):**1.**

T_agent: “leopar kelimesinin tanımıyla ilgili sorular sorayım mı?(e/h)
 l: **e.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:
 somut soyut
 > **somut.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:
 canlı cansız
 > **cansız.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:
 katı sıvı gaz
 > **katı.**

tr_morph_entry(AdKök,[[e,t],[type(noun), sem(_G5324^et(_G5324))], [sm,cns,kt]],
 _, _, _, e, t, ok).

T_agent:et ne demek? Anlatır mısın? (What does “et” mean? Can you explain?)
 l: **et bir yiyecektir. (Et is a kind of food)**

semantic formula: singthe(_G5610, et(_G5610), yiyecek(_G5610))
 yiyecek(_G5610):-et(_G5610).

T_agent:Teşekkürler, öğrendim... (Thanks, I learned).

semantic formula: plurall(_G6715, leopar(_G6715), vahşi(_G6715))
 vahşi(_G6715):-leopar(_G6715).

T_agent:Teşekkürler, öğrendim... (Thanks, I learned).

Figure 5.4. Sample dialogue (continued)

```

semantic formula: singthe(_G10498, leopar(_G10498), singsome(_G10999,
et(_G10999), beslen(_G11628, _G10498, _G10999, [_G11637, _G11640,
_G11643, _G11646], aorist, none, pos, generic)))
leopar(x(1))
id_table(x(1), _G13343^leopar(_G13343))
et(x(2))
id_table(x(2), _G13251^et(_G13251))
beslen(_G11628, x(1), x(2), [_G11637, _G11640, _G11643, _G11646], aorist,
none, pos, generic)
T_agent: Teşekkürler, öğrendim...

semantic formula: singthe(_G8217, (küçük(_G8217), çocuk(_G8217)),
singthe(_G8576, leopar(_G8576), gör(_G9046, _G8217, _G8576, [_G9055,
_G9058, _G9061, _G9064], narrative_past, none, pos, generic)))
küçük(x(3)).
id_table(x(3), _G9670^küçük(_G9670)).
çocuk(x(3)).
id_table(x(3), _G9775^çocuk(_G9775)).
leopar(x(4)).
id_table(x(4), _G9572^leopar(_G9572)).
gör(_G9046, x(3), x(4), [_G9055, _G9058, _G9061, _G9064], narrative_past,
none, pos, generic).
T_agent:Teşekkürler, öğrendim... (Thanks, I learned).

```

Figure 5.5. Sample dialogue (continued)

In the example, the word “et” (meat) is used in one of the descriptive sentences and it has no entry in the lexicon. So TOYagent wants the user to define that word before processing the descriptive sentence. When an entry is added for “et” and its descriptive sentences are processed, the agent returns to the first sentence “küçük çocuk leoparı görmüş” (the little child has apparently seen the leopard) and processes this one.

5.2. DCG Parsing of a Sentence

If all the words are morphologically checked, the sentence is sent to the `get_sem(Wordlist,Sem,Type,Statement)` predicate where `Wordlist` is the list that contains the words in the input line. This predicate performs the DCG parsing of the input. If conjoined sentences are given to the program, they are sent to the `tsc(Sem,[],Wordlist,[])` predicate and the semantic formulas of the conjunct sentences are inserted into the `Sem` argument. If a single sentence is given, it is sent to the `ts(Statement,_,Sem,Wordlist,[])` predicate for this purpose. The `Type` argument shows the type of the sentence, 1 for single sentences and 2 for conjoined sentences. The `Statement` argument shows the mood of the sentence(s).

TOY and TOYagent implement sentence analysis by defining some Turkish phrase structure rules, methods used for generating the semantic representation of the sentence using these rules, and control mechanisms used in these methods.

Subject, object and verb are main constituents of Turkish sentences, where they can be used in any order. Changes in the position of a phrase do not change its grammatical function in a sentence. Basically, the grammatical function is determined according to the case of the word, and this causes Turkish to have free word order

Canan kahvaltıda çayı içti (Canan drank tea at breakfast)
 Subject +Location +Object +Verb
 Canan çayı kahvaltıda içti. (At breakfast, Canan drank tea)
 Subject +Object +Location +Verb

In an application where defining rules for all Turkish sentence types is not aimed, a restriction can be imposed on the number of implemented phrase structure rules. In TOY and TOYagent, this restriction is made by adopting target scenarios for the program at the outset. The TOY syntax infrastructure allows the addition of new syntax rules, and we used this capability in TOYagent. So, in TOYagent we do not cover the word order possibilities in Turkish other than the ones we defined by DCG rules. These rules are given in Figure 5.6.

Each grammar rule is represented in the DCG formalism. DCG rules can be automatically converted to corresponding Prolog rules. Each DCG rule has a left-hand side and a right-hand side. The left-hand side corresponds to the phrase that can be decomposed into the clauses on the right-hand side. This left-hand side is converted to the head and the right-hand side is converted to the body of the corresponding Prolog rule.

Turkish syntax rule:

“A noun phrase can consist of a determiner and a noun”

Standard linguistic notation:

np → determiner noun

The DCG rule:

```
tnp(Case,Sem,N,3) --> td(N,(X^Res)^Sem,_),
                    tn(Case,X^Res,N,3).
```

Prolog:

```
tnp(Case,Sem,N,3,List1,List3):- td(N,(X^Res)^Sem,List1,List2),
                               tn(Case,X^Res,N,3,List2,List3).
```

In this Prolog rule, List1 is the original input string, such as [bir, çocuk], List2 is the input string without the determiner, such as [çocuk], and List3 is the input string without the initial determiner and the noun, such as [].

The rules added for TOYagent are shown in bold in Figure 5.6. The rules that we define may not be identical with the rules defined by linguists like the rule “A noun phrase can consist of a determiner and a noun” .

SC → S S1
S1 → CONJ SC ε
S → NP VP
NP / NP_1 → ADJ D N/ N1
NP / NP_1 → D ADJ N/N1
NP / NP_1 → PARTICIPLE D N/N1
NP / NP_1 → D PARTICIPLE N/N1
NP / NP_1 → D N/N1
NP → PR
NP → IDPR
NP → PN
NP → N NX
NX → N
NX → N NX
NP → NP CONJ NP
NP → QP
NP → QP NP
NP → [Dur, Unit]
NP → [Unit, Begin]
NP → QP TU
PARTICIPLE → NP V1
VP → NP V
VP → QP V
VP → NP_1 V
VP → V
VP → NP_1 (replaces VP → NP rule in TOY)
VP → ADJ_1

Figure 5.6. Turkish phrase structure rules used in the thesis

Four important changes have been made in TOYagent regarding the syntax. First, we added a rule for handling conjoined sentences. $SC \rightarrow S S1$ is the first rule to be processed. S1 is a recursively defined structure that matches any number of sentences preceded by

conjunctions. Presently, only “ve” (and) and “ise” (if) are accepted as top-level conjunctions. The treatment of “veya” (or) in sentences whose top-level conjunction is “ise” will be explained in Section 5.4.2. If “ve” (and) is the main conjunction, then the conjoined sentences are processed independently. But if “ise” (if) is used, then they are processed together:

“Ahmet küçük bir çocuk ise o kahvaltıda süt içmeli.” (If Ahmet is a little child, then he must drink milk at breakfast.)

The two conjunct sentences must be taken together while processing, because the overall sentence meaning is not a straightforward concatenation of the individual phrase meanings. In particular, the overall sentence does not say that Ahmet *is* a little child.

Another problem in TOY was that, nominal verbs formed of adjectives, like “güzelidir” (is beautiful), were not recognized. We corrected this by the addition of the required syntax rule to the program.

Our third addition is the treatment of participle clauses, which TOY lacked. In the sentence, “Didem çocuğu olan bir bayandır” (Didem is a woman who has a child), “çocuğu olan” (who has a child) is a participle clause. In TOYagent, a participle clause consists of a verb marked with the appropriate participle clause suffix (-yAn or -DIK) and the noun phrases associated with that verb, with an implicit gap corresponding to the noun phrase coreferential with the head noun:

“**Fırında pişen** ekmek lezzetliydi.” (The bread baked in the oven was delicious)

Here, “fırında pişen” (the one baked in the oven) is the participle clause. “fırında” (in the oven) is accepted as a noun phrase and “pişen” (baked) is accepted as the verb of the participle clause.

Any noun phrase or a verb that is already accepted by TOYagent can be used as a constituent of a participle clause. But the verb of the participle clause must not have any time or person suffixes.

5.2.1. Noun Phrases (NP and NP_1)

Noun phrases may refer to concrete or abstract entities. For a detailed breakdown of the NP rules in TOYagent, see Table 5.2.

Table 5.2. Noun phrase structures

NP (NP_1 +) Constituents	Samples
Proper Noun	Ahmet
Noun Noun_1 (+)	ekmek(bread), ben(I),... (Çocuktur – he/she is a child)
Noun(s)	Ahmet,ben (I)
Determiner + Noun Determiner + Noun_1(+)	Bir çocuk(A child) (bir çocuktur- he/she is a child))
Determiner+Adjective+Noun Determiner+Adjective+Noun_1 (+) Determiner + Participle + Noun Determiner + Participle + Noun_1 (+)	Bir küçük çocuk (a little child) (Bir küçük çocuktur - he/she is a little child) (Çocuğu olan bayan-the woman who has a child) (Çocuğu olan bayandır – she is the woman who has a child)
Adjective+Determiner+Noun Adjective+Determiner+Noun_1 (+) Participle + Determiner + Noun Participle + Determiner + Noun_1 (+)	Küçük bir çocuk (a little child) (Küçük bir çocuktur – he/she is a little child) (Çocuğu olan bir bayan- A woman who has a child) (Çocuğu olan bir bayandır- She is a woman who has a child)
Noun P.+Conj+Noun P.	Ahmet ve çocuk (Ahmet and the child)
Question P.	Hangi (Which)
Question P.+Noun P.	Hangi ekmek (Which bread)
Duration +Time Unit/ Time Unit + Begin	4 gün(4 days)/saat 10’ da (at 10 o’clock)
Question P. + Time Unit	Kaç saat (How many hours)

The NP structures added for TOYagent are shown in bold in Table 5.2.

Each NP structure has a number of arguments (in addition to the usual ones representing the surface form and semantic formula) in the Prolog implementation. These arguments are used for imposing additional linguistic constraints during syntactic analysis. As an example, consider the following sentences:

“Didem annelerdir” (Didem are mothers)

“Didem bir annedir” (Didem is a mother)

The subject is singular but the verb is plural in the first sentence. Both of them are singular in the second one. The former is syntactically incorrect and must be rejected by the syntactic parser. This can be done by forcing the number arguments of the noun phrase and verb phrase to match.

The grammatical function of a word in a sentence is determined according to the case of the word, as explained in Section 5.2. Case is therefore one of the arguments of NP structures. In TOYagent, the mood argument has been added to NP structures. This indicates the mood of the sentence if an NP is used as a verb.

5.2.2. Verb Phrases (VP)

Verb phrases refer to the event or actions described in the sentence. For a detailed breakdown of the VP rules in TOYagent, see Table 5.3.

Table 5.3. Verb phrase structures

VP Constituents	Samples
NP_1	(Çocuktur – he/she is a child)
Adj_1	(Güzeldir – he/she is beautiful)
Verb	Uyudu (he/she slept)
Noun P.+ Verb	Ekmek yedim (I ate bread)
Question P.+ Verb	Kim geldi (Who came)
Noun P.+Noun P.+Verb	Çocuk okula geldi (The child came to the school)

The VP structures added for TOYagent are shown in bold in Table 5.3.

Each VP structure has five arguments:

- Mood: either statement, or ynoq (yes/no question), or wh-question (which, what, where, etc.)
- Sense: either positive or negative
- Number: whether the VP is singular or plural
- Person: either 1, 2, or 3
- Sem: semantic formula of the sentence

As explained in Section 4.2.1.8, different morpheme entries may exist for a verb. In order to handle multiple variants of the same verb with different semantic representations, some existing tvp rules were modified and new ones were added. This is the fourth major change realized in TOYagent.

$\text{tvp}(\text{ST1}, \text{ST2}, \text{ST3}, \text{N}, \text{P}, \text{X}^{\wedge}\text{Pred}) \rightarrow \text{tv}(\text{ST1}, \text{ST2}, \text{ST3}, \text{N}, \text{P}, \text{T}^{\wedge}\text{L}^{\wedge}\text{Y}^{\wedge}\text{X}^{\wedge}\text{Pred})$.

(exists in TOY)

$\text{tvp}(\text{ST1}, \text{ST2}, \text{ST3}, \text{N}, \text{P}, \text{X}^{\wedge}\text{Pred}) \rightarrow \text{tv}(\text{ST1}, \text{ST2}, \text{ST3}, \text{N}, \text{P}, \text{T}^{\wedge}\text{L}^{\wedge}\text{X}^{\wedge}\text{Pred})$.

(added in TOYagent)

The first rule only accepts transitive verbs, as in “Kemal kahvaltıda peynir yedi” (Kemal ate cheese at breakfast). The second one provides a way of accepting intransitive verbs, as in “Kemal uyudu” (Kemal slept).

5.2.3. The Semantic Formula of a Sentence

The semantic formula of a sentence is basically a (possibly nested) FOPC expression (see Section 3.2) in Prolog. This expression is constructed during the DCG parsing of a sentence. The parsing of the sentence starts from the $S \rightarrow NP VP$ rule. (see Figure 5.7) The expression is constructed by using the semantic formulas of the sentence constituents. If the constituent is a noun (property), a quantified expression for this is used in the expression. The determiner of the noun plays an important role in the determination of the quantified expression, which is explained in the following section.

5.2.3.1. Quantifiers and Determiners. Quantifiers in a formal logic representation are counterparts of determiners in natural language. Each determiner in a sentence corresponds to a quantifier in formal logic notation.

In “Bütün anneler güzeldirler ve melektirler” (All mothers are beautiful and angels), ‘bütün’ is the determiner of ‘anne’ and corresponds to the ‘ \forall ’ (all) quantifier in formal logic notation. More than one determiner can correspond to the same quantifier. Consider the following pair of sentences:

Çocuk bir elma yiyor (The child is eating an apple)

Çocuk elma yiyor (The child is eating an apple)

In the first sentence, “bir” (a) is used as the determiner of the apple but in the latter one, a null determiner (one with no surface component) is used. In Turkish both of them have the same meaning, “an apple”, and are represented by the same quantifier.

Each determiner is defined by a Prolog rule in TOY:

```
td (sing,(X^Res)^(X^Sco)^singall(X,Res,Sco),before) --> [her].
```

X is the quantified variable, Res is the semantics of the noun phrase preceded by this determiner, Sco is the predicate that is supposed to be true about the quantified variables, and singall(X,Res,Sco) is the Prolog formula that will be constructed by these components. Ten such determiners are defined in TOY:

```
singsome(X,Res,Sco) --> [bir] ; [].
```

```
singthe(X,Res,Sco) --> [].
```

```
plursome(X,Res,Sco) --> [bazı] ; [].
```

```
singall(X,Res,Sco) --> [her].
```

```
plurall(X,Res,Sco) --> [bütün].
```

```
plurthe(X,Res,Sco) --> [].
```

```
which(X,Res,Sco) --> [hangi].
```

```
how_many(X,Res,Sco) --> [kaç].
```

Although “some” can be used for singular (someone) and plural (some books) nouns in English, it has different counterparts for singular (‘biraz’ or null) and for plural (‘bazı’) nouns in Turkish. Consequently, a distinction between them is implemented by defining two different predicates for each one: singall, singsome, plurall, plursome.

5.2.3.2. Construction of a Sample Sentence’s Semantic Formula. In Figure 5.7, the semantic formula of the sentence “Ehliyeti olan bir kişi araba kullanır” (Somebody who has a driving license drives a/any car) and the DCG rules used during the construction of the formula are given. Figure 5.8 shows the semantic formulas and the quantified expressions used in this semantic formula.

<p><u>DCG rules used during the construction:</u></p> <p>S (Ehliyeti olan bir kişi araba kullanır) → NP (ehliyeti olan bir kişi) VP (araba kullanır)</p> <p>NP (ehliyeti olan bir kişi) → PARTICIPLE (ehliyeti olan) D (bir) N(kişi)</p> <p>PARTICIPLE (ehliyeti olan) → NP (ehliyeti) V1 (olan)</p> <p>NP (ehliyeti) → D() N (ehliyeti)</p> <p>VP (araba kullanır) → NP (araba) V (kullanır)</p> <p>NP(araba) → D() N (araba)</p> <p><u>Semantic Formula:</u></p> <p>singsome(X,(singsome(Y,ehliyet(Y),var(EvNo,X,Y,[B,E,D,U],_,_,_)), kişi(X)),singsome(Z,araba(Z), kullan(EvNo2,X,Z,[B2,E2,D2,U2],_, none, pos, generic)</p>

Figure 5.7. DCG rules used in the construction with the semantic formula

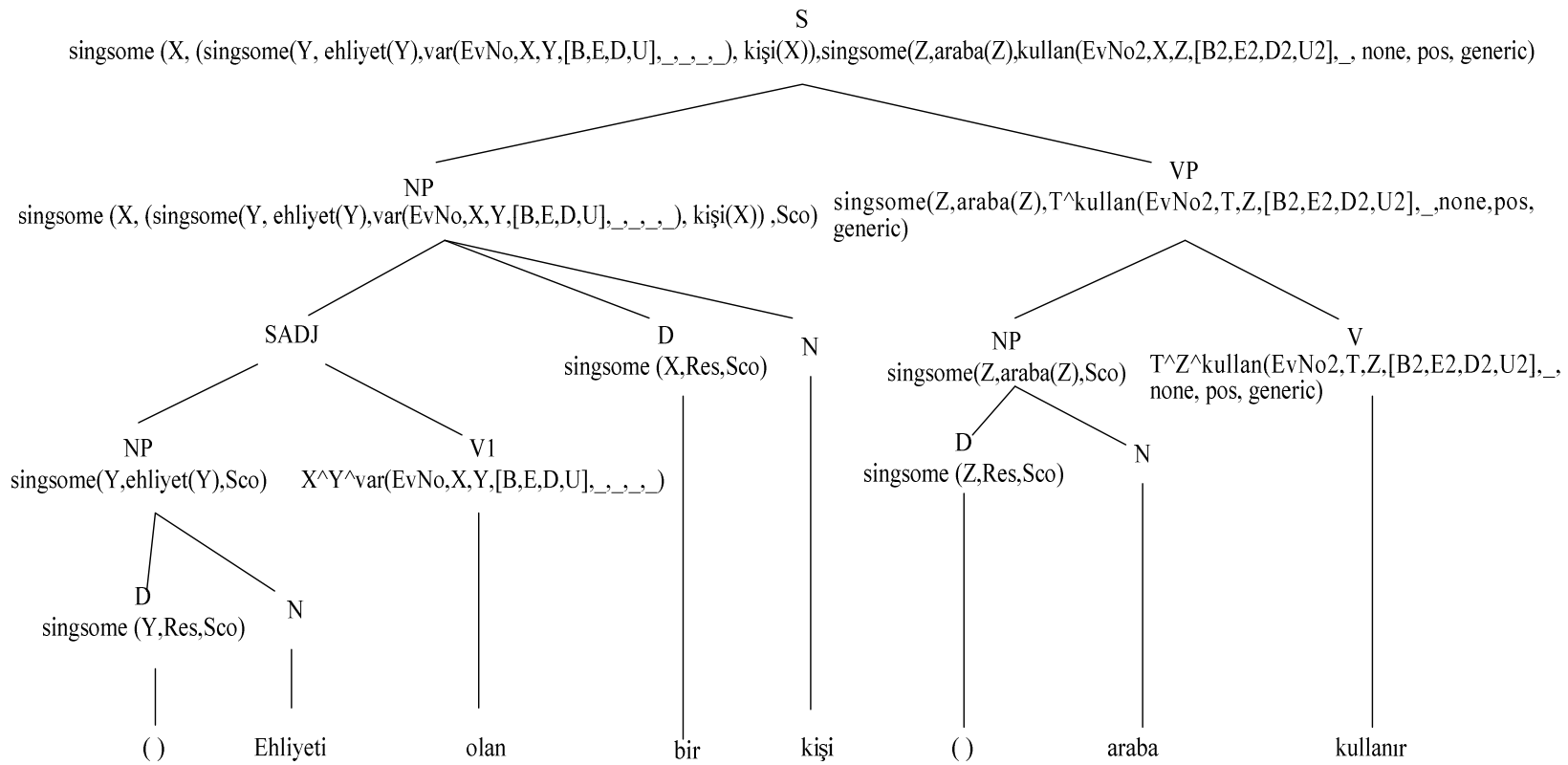


Figure 5.8. Semantic formula of the sentence “Ehliyeti olan bir kişi araba kullanır”

As seen in Figure 5.8, each noun is placed in a quantified expression. The expression `singsome(Y, ehliyet(Y), var(EvNo,X,Y,[B,E,D,U],_,-,-,-))` is used for the word “ehliyet” (the driving license).

5.3. Modifications before Semantic Transformation

In TOY, the semantic formulas retrieved from the “get_sem” predicate are sent to the “inputorsoh(Statement,Sem,TS,Type)” predicate. If the mood of the sentence is ‘statement’, then two modifications can be done on the semantic formula before it is transformed to Prolog predicates and rules. One of these is related to the conjunction “hariç” (except), and the other deals with negative knowledge. In TOYagent, a new modification for anaphora resolution has been added.

5.3.1. Removal of the Conjunction “Hariç” (Except)

The semantic formula of a sentence which contains noun phrases that are combined with the “hariç” (except) conjunction is rewritten into a “plurall” predicate that contains a negation in its restrictor, and the transformation rule that is applied to “plurall” predicates is subsequently applied to this representation.

5.3.2. Negative Knowledge Modification

There is a discrepancy between the logical formula of a natural language sentence with negative sense and the negation of the logical formula of the positive version of the same sentence [1]. TOY handles this problem with the “convert (Sem, NewSem)” routine, which changes the outermost “plurall” predicate, if any, to “singsome” predicate or the innermost “singsome” predicate, if any, to “plurall” predicate if the sense of the sentence is negative.

Details of both of this modification and conjunction removal are given in [1].

5.3.3. Anaphora Resolution

We developed our own personal pronoun resolution method. Our aim was to implement a realistic method. As mentioned in Section 2.3, human beings resolve anaphors by using the commonsense knowledge they have and hints from the discourse in general, but it seems to us that they never use some of the techniques used in methods described in Section 2.3 like statistical information. In TOYagent, the method implemented for personal pronoun resolution is based on the usage of commonsense knowledge so our method can be seen as a traditional approach. The commonsense knowledge is represented by the semantic network which was explained in Section 3.4. In TOYagent, cataphora resolution is not implemented; our resolution method only works backwards.

5.3.3.1. Resolution Method. If a personal pronoun is used in the input sentence, its semantic representation that was looked up from the lexicon is placed into the representation of the sentence by the DCG parser. But, as explained above, each anaphor refers to certain objects like places, people, etc. in the discourse. So, the anaphor must be replaced with the referent in the representation, as given in Figure 5.9.

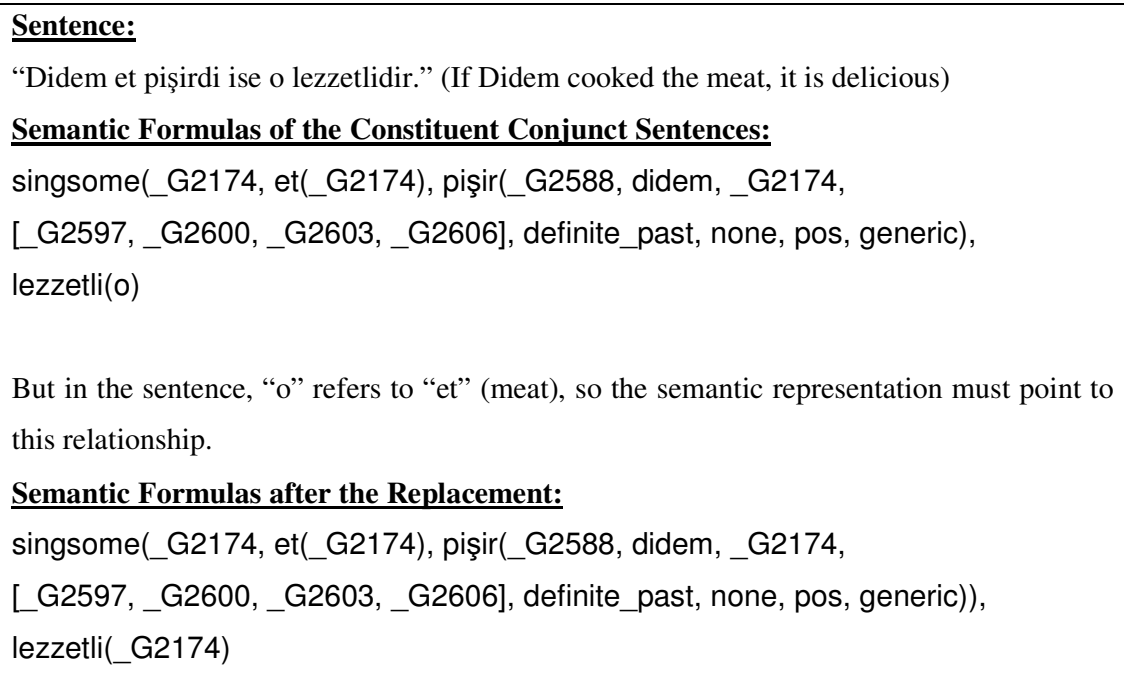


Figure 5.9. Replacement of an anaphor in a semantic formula

The third modification method, anaphora resolution, performs this replacement. But, if a referent to an anaphor is not found in the discourse, then the anaphor remains in the semantic formula.

The anaphora resolution method is called when the given sentence is a conjoined sentence and the discourse where the referent will be searched is assumed to be the combination of the conjunct sentences preceding the sentence where the anaphor resides. As mentioned above, cataphora resolution is not implemented in TOYagent, so there is no reason to take the succeeding sentences into account. If a conjoined sentence is given to our program, each conjunct sentence is parsed sequentially. As a predictive strategy, a descriptonal list of each theta item in these sentences is inserted into a possible referents list after successful parsing. The referents list formed during the processing of the sentence in Figure 5.9 until the word “o” (it) is:

Referents List = [[1, propernoun, didem, [[sm, cn, i], [ek(6)]], agent], [1, noun, et, [[sm, cns,kt], [ek(6)]], theme]]

Each member of this referents list is a list itself. Each sublist consists of five members.

- **First member:** number of the conjunct sentence containing this word
- **Second member:** syntactic category of the word
- **Third member:** the stem of the word
- **Fourth member:** the value of the second output list retrieved when the word is sent to the “parse” predicate, as explained in Section 4.3. This member plays an important role in resolution.
- **Fifth member:** theta role of the word

When an anaphor is encountered in a conjunct sentence, the members of this list are searched.

Five steps are followed during the resolution process, if an anaphor is encountered in a sentence. At the first step, the theta role that the anaphor carries in the sentence is

determined. In our present implementation of syntax, if the verb of this conjunct sentence is nominal, the only role that can be carried by the anaphor is agent, otherwise, this can be either agent, or theme, or location. Therefore, in Figure 5.6, the word “o” (it) is the agent of the second sentence.

Each verb’s morpheme entry indicates the semantic classes and distinctive features of the possible agents, themes, and locations that can be the constituents of a sentence where this verb is used. Each adjective’s morpheme entry indicates the semantic classes and distinctive features of the possible agents (nouns) which can reasonably be modified by this adjective. Each noun’s morpheme entry indicates the semantic classes and distinctive features of the noun (see Section 4.2.2). At the second step, if the verb is nominal, all of these indicators are retrieved, otherwise, the indicators which have the same theta role caption with the role that the anaphor carries are selected. In Figure 5.6, the verb is nominal, so all of the indicators in the morpheme entry are taken.

[[sm,cns,sv],[sm,cns,kt]]: Possible agents of the word “lezzetli”

At the third step, the program scans all members of the referents list and tries to match the indicator in hand with the first sublist of the fourth member of these members. If more than one indicator is defined for the same role as seen in the word “lezzetli”, all of these indicators are tried sequentially. This process continues until a match is found or all indicators in hand are checked.

[1, noun, et, [[**sm,cns,kt**], [ek(6)], theme] list and the indicator [**sm,cns,kt**] are matched.

If an indicator is matched, at the fourth step, the second member of the matched sublist in the referents list is extracted. If this member is “noun,” the program has to obtain the variable name representing this noun in the semantic formula, to use the same name in place of the anaphor. The word “et” (meat) is used in the first conjunct sentence, the order of the sentence is obtained from the first member of the matched sublist. The theta role of the word is “theme,” which is obtained from the fifth member of the matched sublist. The agent extracts the variable used in that role of the first conjunct sentence’s semantic

formula, “_G2174” in this case. If the second member is “propernoun”, the agent extracts the third member of the matched sublist.

At the final step, the proper noun constant or the variable that is found at the fourth step is replaced with the anaphor.

The algorithm of the anaphora resolution method is given in Figure 5.10.

1. **Find the theta role of the anaphor in the sentence:**
 - a. If the verb is nominal, the theta role is “agent” .
 - b. else, find in which slot of the semantic formula (agent, theme, location) this anaphor is used and set it to the theta role.
2. **Retrieve the indicator(s) from the verb’s morpheme entry:**
 - a. If the verb is nominal, take all of the indicators in the morpheme entry.
 - b. else, take all of the indicators which have the same theta role with the anaphor’s role.
3. **Try to match the anaphor with a referent:**
 - a. While all of the indicators are not checked and no match is found
 - I. Take one of the indicators
 - II. Compare the indicator with all members of the referents list
4. **Replace the anaphor with the referent if found:**
 - a. If a referent is found
 - I. If the syntactic category of the referent is “noun”, extract the variable representing this referent from the semantic formula where it is used.
 - II. If the syntactic category of the referent is “propernoun” extract the referent from the semantic formula where it is used.
 - III. Replace the anaphor with the variable or with the propernoun and return the new semantic formula
 - b. else, return the same semantic formula

Figure 5.10. Algorithm of the anaphora resolution method

As for the shortcomings of this method, the referent may be in any one of the preceding sentences in the discourse. In some cases it is reasonable to start the search from the closest sentence, but in some cases it is not. So there is no clear distinction about which approach is the correct one. We decided to let our program always start the search from the first conjunct sentence.

There may be multiple candidate referents for an anaphor in the discourse. But if one is found during the scanning operation, it is accepted as the referent and the scan is ended. This may cause some incorrect matches, as given in Figure 5.11

Türkçe Okur Yazar (Literate in Turkish)
 Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)

Cümleniz> **Didem çocuğu azarladı ise o yaramazlık yapmıştır. (If Didem has scolded the child, he/she must have misbehaved)**

yap(_G5062, didem, _G4506, [_G5071, _G5074, _G5077, _G5080], narrative_past, none, pos, generic):-azarla(_G3546, didem, _G3070, [_G3555, _G3558, _G3561, _G3564], definite_past, none, pos, generic), yaramazlık(_G4506), çocuk(_G3070).

T_agent: Teşekkürler, öğrendim...

Figure 5.11. Multiple candidate referents example

In Figure 5.11 an anaphor is used as the agent of the second conjunct sentence and the agent indicator of the verb is [sm, cn, i]. Both Didem and “çocuk” (child) are human beings and satisfy the conditions to be the referent of this anaphor. The agent initially finds Didem as the referent and ends the search operation. But the referent of the anaphor shown in Figure 5.11 is “çocuk” (child), not Didem. The anaphora resolution method must be improved and the semantic network must be more detailed to find the correct referent in such cases.

When necessary changes are implemented for handling possessive suffixes in TOYagent, the anaphora resolution method must be improved because using the current resolution method, the referent of the anaphor in the sentence “Onun ağabeyi mühendistir” (His/Her brother is an engineer) can not be found.

5.4. Incorporation of Knowledge

After the postprocessing described in the previous section, the modified semantic formulas are transformed to Prolog predicates and rules. Finally, these are inserted to the knowledge base. If the extracted piece of knowledge is seen to be already present, TOYagent outputs “Biliyorum” (I know), otherwise it outputs “Teşekkürler, öğrendim” (Thanks, I learned).

If the mood of the sentence is a ‘yes/no question’, then the semantic formula is sent to the “search_knowledge(Sem)” predicate and an answer is formed for this question, as will be explained in Chapter 6.

If a conjoined sentence is given to TOYagent, the transformation process on conjunct sentences is implemented by taking the conjunctions between them into account, as will be explained in Section 5.4.2.

The additions to the knowledge base done during a conversation session can only be used in that session.

5.4.1. Simple Sentence Transformation

Different transformation rules are implemented according to the type of the quantifier used in the semantic representation: Existential, Universal, and Definite.

5.4.1.1. Singsome Expressions (Existential Quantifier). If a determiner that causes the expression format (singsome(X, Res, Sco)), which corresponds to the existential quantifier in FOPC, has been used, Prolog facts are created for Res and Sco and inserted to the knowledge base as:

Res.

Sco.

Furthermore, the *Skolemization* process is applied on the expression and the existential quantifier is removed. In this process, each variable, e.g. X in the above representation, is unified with a function, essentially providing a “name” for an object having the properties described existentially quantified formula.

This name is a unique identifier, which is called the *discourse marker* of the thing being mentioned. In TOY, each discourse marker has the form x(No). The “No” value is generated by the program when needed. This notation allows the program to keep track of the distinctions among the entities mentioned in the discourse. Details of the Skolemization process are explained in [1].

The discourse marker assigned in the Skolemization process and a semantic formula of its referent is stored in the knowledge base as “identity table” facts, in the format

id_table(Marker, Sem).

In TOYagent these facts are used in different modules, one of which is the sentence generation from the semantic formula module that will be explained in Section 6.3. An example singsome expression is given in Figure 5.12 where x(1) is the discourse marker given to “araba” (car).

Sentence:

“Ali araba kullanır” (Ali drives a car)

Semantic Formula:

singsome(_G1521, araba(_G1521),kullan(_G1986, ali, _G1521, [_G1995, _G1998, _G2001, _G2004], _G1990, none, pos, generic))

Internal Representation in the Knowledge Base:

araba(x(1)).

id_table(x (1), _G2480^araba (_G2480)).

kullan(_G1986, ali, x (1), [_G1995, _G1998, _G2001, _G2004], _G1990, none, pos, generic).

Figure 5.12. Singsome expression example

5.4.1.2. Plurall Expressions. If a determiner that corresponds to the plurall(X, Res, Sco) expression, which corresponds to the universal quantifier in FOPC, is used, a Prolog rule is created and inserted to the knowledge base:

Sco :- Res.

An example plurall expression is given in Figure 5.13.

<p><u>Sentence:</u> “Bütün çocuklar melektirler” (All children are angels)</p> <p><u>Semantic Formula:</u> plurall(_G1180, çocuk(_G1180), melek(_G1180))</p> <p><u>Internal Representation in the Knowledge Base:</u> melek(_G1180):-çocuk(_G1180).</p>

Figure 5.13. Plurall expression example

5.4.1.3. Singthe and Plurthe Quantifiers. Definite phrases are transformed to expressions with the format singthe(X,Res,Sco) or plurthe(X,Res,Sco), where X refers to a definite object. A definitely referenced noun phrase can be introduced earlier in the conversation, or assumed to be a part of the background knowledge of the hearer. Discourse markers are again used before the insertion to the knowledge base. Although different discourse markers are given to existentially quantified noun phrases for each of their appearances in the sentence, the same discourse marker must be used for a definitely quantified noun phrase for each of its appearances, as implemented by [8].

An example singthe expression is given in Figure 5.14.

<p><u>Sentence:</u></p> <p>Canan çayı içti. (Canan drank the tea)</p> <p><u>Semantic Formula:</u></p> <p>singthe(_G2458, çay(_G2458), iç(_G2910, canan, _G2458, [_G2919, _G2922, _G2925, _G2928], definite_past, none, pos, generic))</p> <p><u>Internal Representation in the Knowledge Base:</u></p> <p>çay(x(1)).</p> <p>id_table(x(1), _G3470^çay(_G3470)).</p> <p>iç(_G2910, canan, x(1), [_G2919, _G2922, _G2925, _G2928], definite_past, none, pos, generic).</p>

Figure 5.14. Definitely quantified object example

If these quantifiers are nested in the semantic representation, rules defined for each quantifier type are applied to the quantifiers of this type iteratively, as given in Figure 5.15.

<p><u>Sentence:</u></p> <p>“bütün çocuklar kahvaltıda süt içerler” (All children drink milk at breakfast)</p> <p><u>Semantic Formula:</u></p> <p>plurall(_G1883, çocuk(_G1883), singthe(_G2216, kahvaltı(_G2216), singsome(_G2697, süt(_G2697), iç(_G3178, _G1883, _G2697, _G2216, [_G3188, _G3191, _G3194, _G3197], _G3183, none, pos, generic))))</p> <p><u>Internal Representation in the Knowledge Base:</u></p> <p>kahvaltı(x(1, _G1883):-çocuk(_G1883).</p> <p>id_table(x(1, _G1883), _G3914^kahvaltı(_G3914)).</p> <p>süt(x(2, _G1883):-çocuk(_G1883).</p> <p>id_table(x(2, _G1883), _G3809^süt(_G3809)).</p> <p>iç(_G3178, _G1883, x(2, _G1883), x(1, _G1883), [_G3188, _G3191, _G3194, _G3197], _G3183, none, pos, generic):-çocuk(_G1883).</p>

Figure 5.15. Nested quantifier example

In the sentence given in Figure 5.15, it is said that “each child drinks his/her own milk at his/her own breakfast.” To indicate this ownership, functions that map from child to milk and from child to breakfast are used during the Skolemization process.

5.4.2. Conjoined Sentence Transformation

If the given sentence consists of conjunct sentences, different transformation rules are necessary for each different conjunction type that may be used between them.

5.4.2.1. “And” (ve) Conjunction. If only (one or more copies of) the “ve” (and) conjunction is used between the conjunct sentences, these sentences are treated independently and each of them are inserted to the knowledge base as explained in Section 5.4.1. (Figure 5.16)

Sentence:

“Kemal araba kullanır ve o iyi bir şofördür.” (Kemal drives a car and he is a good driver)

Semantic Formulas:

singsome(_G2913, araba(_G2913), kullan(_G3393, kemal, _G2913, [_G3402, _G3405, _G3408, _G3411], _G3397, none, pos, generic)),
(iyi(kemal), şoför(kemal)).

Internal Representations in the Knowledge Base:

araba(x(1)).

id_table(x(1), _G5843^araba(_G5843)).

kullan(_G3393, kemal, x(1), [_G3402, _G3405, _G3408, _G3411], _G3397, none, pos, generic).

iyi(kemal).

şoför(kemal).

Figure 5.16. ‘ve’ (and) conjunction example

Any number of sentences can be combined using the “ve” conjunction.

<p><u>Sentence:</u></p> <p>“Ali yaramazlık yapmıyor ise o uslu bir çocuktur.” (If Ali does not misbehave then he is a well-behaved child)</p> <p><u>Semantic Formulas:</u></p> <p>singsome(_G2793, yaramazlık(_G2793), yap(_G3273, ali, _G2793, [_G3282, _G3285, _G3288, _G3291], progressive, none, neg, generic)), (uslu(ali), çocuk(ali))</p> <p><u>Internal Representations in the Knowledge Base:</u></p> <p>uslu(ali):-yap(_G3273,ali,_G2793,[_G3282,_G3285,_G3288,_G3291], progressive, none, neg, generic), yaramazlık(_G2793).</p> <p>çocuk(ali):-yap(_G3273,ali,_G2793,[_G3282,_G3285,_G3288,_G3291], progressive, none, neg, generic), yaramazlık(_G2793).</p>
--

Figure 5.18. ‘ise’ (if) conjunction sample-2

5.4.2.3. Mixed Conjunctions (‘ise’ (if) – ‘ve’ (and) – ‘veya’ (or)). If the sentence at the left side of the “ise” (if) conjunction is a conjoined sentence, where the conjunction “ve” (and) or “veya” (or) is used, Prolog rules are created and added to the knowledge base. One rule is created for the sentence at the right side of the “if” conjunction or two rules are created if the verb is a noun phrase consisting of an adjective and a noun. The heads of such rules are formed of the verb fact of the sentence at the right side.

If the “ve” (and) conjunction is used on the left side, the remaining facts of the sentence at the right side are combined with the facts of the sub sentences at the left side. This combination forms the body of the rule(s), as given in Figure 5.19.

If the “veya” (or) conjunction is used on the left side, the remaining facts of the sentence at the right side are combined with the facts of each conjunct sentence at the left

side separately. These combinations are separated with “;” (‘or’ indicator in Prolog) and form the body of the rule, as given in Figure 5.20.

Sentence:

“Canan kahvaltıda peynir yedi ve o bitti ise o doymuştur.” (If Canan ate cheese at breakfast and it ended, she must be full) semantic formula:

Semantic Formulas:

singthe(_G4259, kahvaltı(_G4259), singsome(_G4746, peynir(_G4746),
ye(_G5206, canan, _G4746, _G4259,
[_G5216, _G5219, _G5222, _G5225], definite_past, none, pos, generic))),

bit(_G6495, _G4746, _G6407, [_G6504, _G6507, _G6510, _G6513],
definite_past, none, pos, generic),

doy(_G7682, canan, _G7558, [_G7691, _G7694, _G7697, _G7700],
narrative_past, none, pos, generic).

Internal Representations in the Knowledge Base:

doy(_G7682, canan, _G7558, [_G7691, _G7694, _G7697, _G7700],
narrative_past, none, pos, generic):-iç(_G5206, canan, _G4746, _G4259,
[_G5216, _G5219, _G5222, _G5225],
definite_past,none, pos, generic),
bit(_G6495, _G4746, _G6407,
[_G6504, _G6507, _G6510, _G6513],
definite_past,none,pos,generic),
kahvaltı(_G4259),süt(_G4746).

Figure 5.19. ‘ve’ (and) and ‘ise’ (if) conjunctions sample

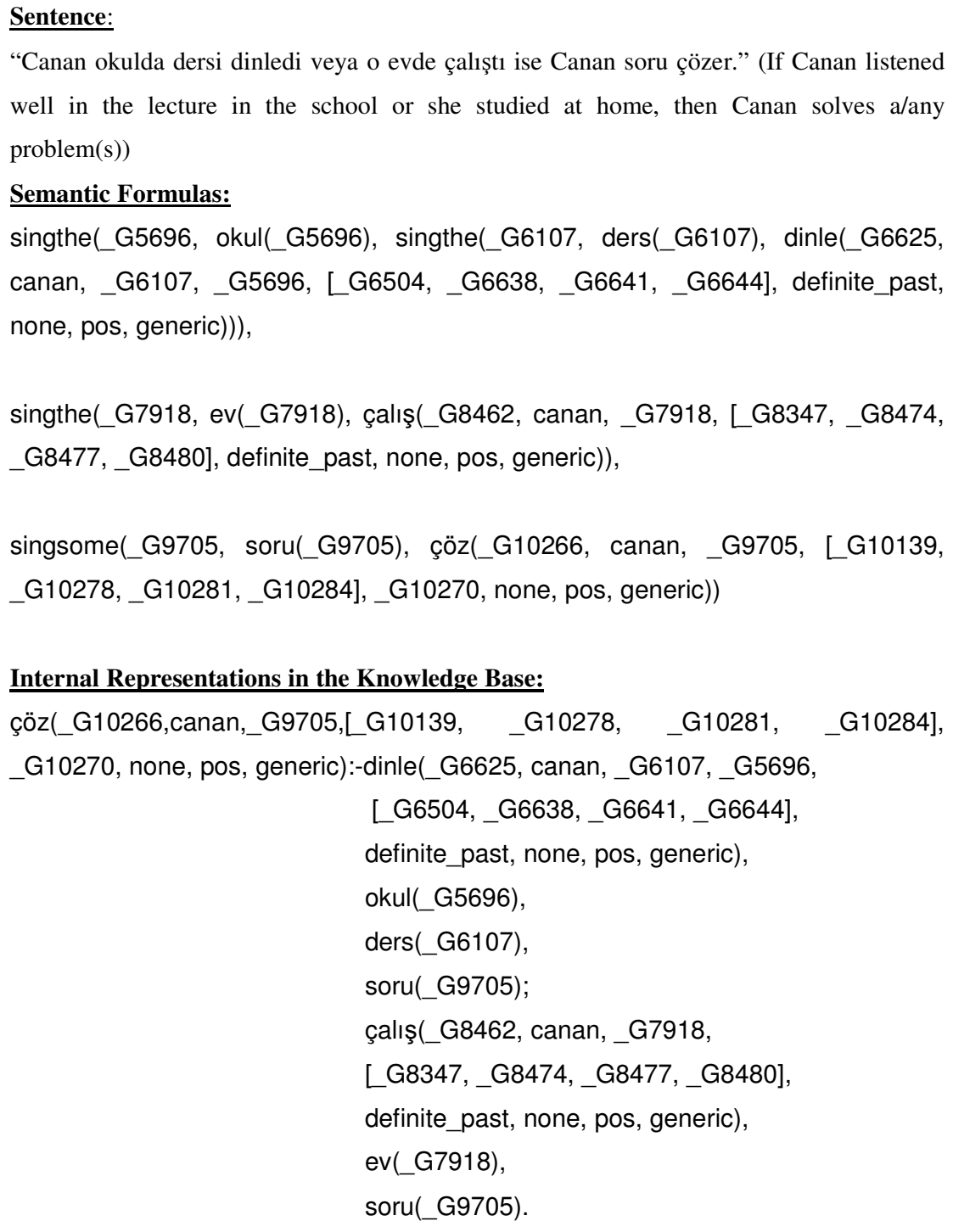


Figure 5.20. ‘veya’ (or) and ‘ise’ (if) conjunctions sample

New conjunction combinations can be added to TOYagent in the future, as long as proper attention is paid to the logical combination rules involved in this process.

5.4.3. Participle Clause Transformation

Participle clauses can be used to describe, much like adjectives, entities represented by following noun phrases. The participle clause “fırında pişen” (the one baked in the oven) is used in the sentence “Fırında pişen ekmek lezzetliydi” (The bread baked in the oven is delicious) to describe the bread. All participle clauses which are accepted by our program are themselves sentences whose verbs are marked with the appropriate participle clause suffix, like the –en suffix in the above sentence. Although both are used to describe other entities, the semantic representation of a participle clause differs from the representation of an adjective, and is similar to a sentence representation as given in Figure 5.21. The part shown in bold is the semantic representation of the participle clause.

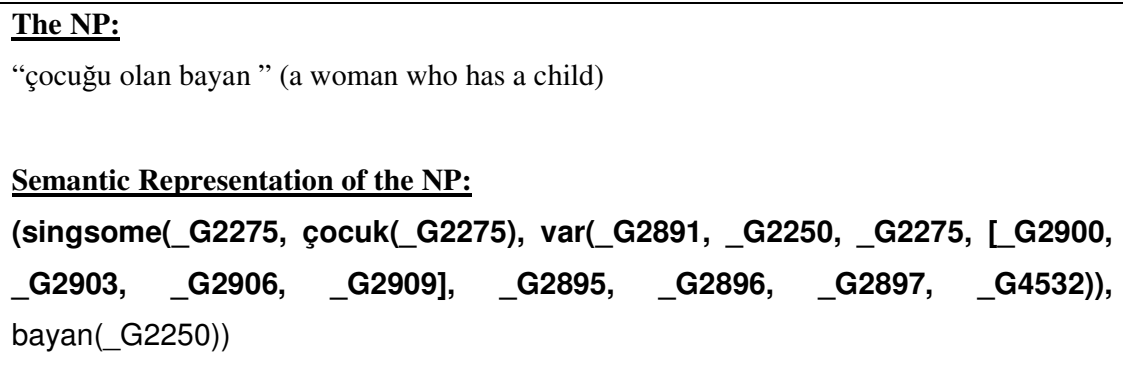


Figure 5.21. Semantic representation of the participle clause “çocuğu olan” (the one who has a child)

The transformation of a participle clause is performed like a sentence and Prolog facts are obtained. These predicates restrict the thing that is described by the participle clause. As an example, in Figure 5.21, the mentioned woman must have a child. So, a sentence where a participle clause is used must be transformed into a Prolog rule and the facts obtained from the participle clause must be added to the body of this rule. The head of this rule is the verb fact of the sentence, and the facts retrieved from the transformation of the sentence must also be added to the body of this rule, as given in Figure 5.22.

<p><u>Sentence:</u> “çocuğu olan bayan annedir.” (The woman who has a child is a mother)</p> <p><u>Semantic Formula:</u> singsome(_G2250, (singsome(_G2275, çocuk(_G2275), var(_G2891, _G2250, _G2275, [_G2900, _G2903, _G2906, _G2909], _G2895, _G2896, _G2897, _G7655)), bayan(_G2250)), anne(_G2250))</p> <p><u>Internal representation in the knowledge base:</u> anne(_G2250):-çocuk(_G2275), var(_G2891,_G2250,_G2275,[_G2900,_G2903,_G2906,_G2909], _G2895,_G2896,_G2897,_G7655), bayan(_G2250).</p>
--

Figure 5.22. Participle clause transformation

5.4.4. Transformation Postprocessing

The “temp_words_content” list, explained in Chapter 4, is emptied when the knowledge contained in the input sentence is asserted to the knowledge base. At that stage, its content is modified and inserted to the “words_content_list”. The words_content list is used in sentence generation from the semantic formula, as will be explained in Section 6.3. Each member of this list has the form:

[discourse_marker given to the word or the proper noun itself, names of the affixed suffixes].

6. QUESTION ANSWERING IN TOYagent

In TOY, the user can query the knowledge base by entering yes/no questions and wh-questions (either who, or what, or which, or when) [1]. None of TOYagent's target scenarios include wh-questions, and no maintenance was done on the codes inherited from TOY for answering these questions. So, although the semantic formula of a wh-question can be obtained in TOYagent, an answer to this can not be formed.

In TOYagent, the user can query the knowledge base by entering only yes/no questions. TOY's standard response mechanism is inherited (see Section 6.1), but two more mechanisms have been added in order to handle questions about uncertain conditions. The three mechanisms used for forming an answer to the given question can be named according to their treatment of the question's semantic formula, as "Same semantic formula" (SSF), "Modified semantic formula" (MSF), and "Deduction on similar semantic formulas" (DOSSF). If the agent can not form an answer using any one of these mechanisms, then it simply outputs "Bilmiyorum" (I don't know). A priority ordering exists between these mechanisms, with SSF having the highest priority, and DOSSF having the lowest priority.

After an answer is formed by the agent using the SSF or the MSF mechanism, the user has the chance of learning how it was computed. If the agent is set to "explanatory mode," then it outputs the information it used during the formation process after it outputs the answer. The outputs are expressed by Turkish natural language sentences. The mode can be changed during the execution. This explanation process is explained in Section 6.3.

If a yes/no question is given to the agent, the semantic formula of this sentence is sent to the "search_knowledge(Sem)" predicate. In order to retrieve the information used during the formation, the "get_reason" predicate is executed.

If “Didem aşçı mıdır?” (Is Didem a cook?) is asked to the agent, it checks whether the semantic formula $aşçı(didem)$ can be proven using the knowledge base. The known facts together with the rule shown in Figure 6.1, enable the successful proof of $aşçı(didem)$. Prolog rules can be used for forming answers only in this mechanism.

6.2. Treatment of Modalities in Question Answering (MSF and DOSSF mechanisms)

Before executing the last two mechanisms, the agent initially extracts verb facts corresponding to the verb in the input sentence from the knowledge base and creates a list for each of them. If that verb is intransitive, then the 8-ary verb facts are extracted and lists consisting of five members are created. The members of such a list are the agent, the location, the tense, the auxiliary tense, and the deontic modality of the extracted verb fact. If the verb is transitive but no location information is given in the input sentence, then the 8-ary verb facts are extracted and lists consisting of five members are created. The members of such a list are the agent, the theme, the tense, the auxiliary tense, and the deontic modality of the extracted verb fact. If the verb is transitive and the location information is given in the input sentence, then the 9-ary verb facts are extracted and lists consisting of six members are created. The members of such a list are the agent, the theme, the location, the tense, the auxiliary tense, and the deontic modality of the extracted verb predicate. These lists are combined to form a general information list, which is the pool of the known pieces of knowledge that may be relevant for answering this question. A sample information list is given in Figure 6.2. After the generation of the general information list, the program also creates a search list for the entered question which has the same format as shown below:

Cümleniz> **Canan kahvaltıda kahve içti mi. (Did Canan drink coffee at breakfast?)**

Search List: [canan, kahve, kahvaltı, definite_past, none, generic]

The last two mechanisms use these lists during the question answering process.

<p>Sentences which have previously been entered:</p> <p>Canan kahvaltıda çay içti. Kemal kahvaltıda çay içti.</p> <p>Verb facts in the KB that include the verb “iç” (drink):</p> <p>iç(_G3071, canan, x(2), x(1), [_G3081, _G3084, _G3087, _G3090], definite_past, none, pos, generic). iç(_G3546, kemal, x(4), x(3), [_G3556, _G3559, _G3562, _G3565], definite_past, none, pos, generic).</p> <p>General Information List:</p> <p>[[canan, çay, kahvaltı, definite_past, none, generic], [kemal, çay, kahvaltı, definite_past, none, generic]]</p>
--

Figure 6.2. Contents of an example general information list

Consider a scenario where if the agent knows that Ali drinks tea at breakfast, and the user asks whether Ali *can* drink tea at breakfast. The first mechanism can not form an answer for this question, because that exact piece of information involving that deontic modality can not be deduced from the first sentence using TOY’s deduction mechanism. We claim that a human being would normally answer this question affirmatively, because the knowledge that a person takes a specific type of action regularly entails the information that person is capable of taking that type of action.

In the MSF mechanism, possible differences in the time and deontic modality arguments between the members of the general information list and the search list are taken into account. In the above example, the difference is in the modalities. Either time or deontic modality or both can be different between the lists. One of four predefined answers can be formed in this mechanism: “Evet” (Yes), “Bilmiyorum” (I don’t know”), “Tahminen, evet” (I guess so), and “İsterse, evet” (If he/she wishes, yes). A priority ordering exists between the predefined answers, with “Evet” (Yes) having the highest priority, and “İsterse, evet” (If he/she wishes, yes) having the lowest priority. The priority is determined according to the level of certainty of the agent about the answer. The formation is based on Tables 6.1 and 6.2.

Each member of the general information list is searched in the MSF mechanism. There may be more than one member that provides the necessary information to form an answer. All formed answers are stored in a list and the one with the highest priority is chosen as the answer.

Table 6.1. Tense and deontic modality table

Tense + Deontic modality	1	2	3	4	5	6	7	8	9	10
1) Definite past+generic	E	B	B	E	T	E	I	B	E	I
2) Future +generic	B	E	B	B	T	B	E	B	B	E
3) Progressive +generic	B	B	E	B	T	B	I	E	B	E
4) Narrative past+ generic	T	B	B	E	T	E	T	B	E	T
5) Aorist + generic	B	B	B	B	E	B	B	B	B	E
6) Definite past+ability	E	B	B	E	T	E	E	B	E	E
7) Future +ability	B	I	B	B	T	B	E	B	B	E
8) Progressive +ability	B	B	E	B	T	B	I	E	B	E
9) Narrative past+ ability	T	B	B	E	T	T	T	B	E	T
10) Aorist + ability	B	B	B	B	I	B	I	B	B	E

Table 6.2. Abbreviations of the answers

Answer	Abbreviation
Evet	E
Bilmiyorum	B
Tahminen evet	T
İsterse evet	I

In Table 6.1, the pairs indexing the rows are the tense and the deontic modality of the known sentence, and the pairs in the columns are the tense and the deontic modality of the question. The lists are matched using this table and the agent outputs the answer using Table 6.2. A sample dialogue is given in Figure 6.3.

Türkçe Okur Yazar (Literate in Turkish)
Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)

Cümleniz> **Ali çay içmiş. (Ali has apparently drunk tea)**
çay(x(1)).
id_table(x(1), _G2280^çay(_G2280)).
iç(_G1801, ali, x(1), [_G1810, _G1813, _G1816, _G1819], narrative_past, none,
pos, generic).
Teşekkürler, öğrendim...

Cümleniz> **Ali çay içer mi. (Does Ali drink tea?)**
T_agent: Tahminen, evet :) (I guess so)

Cümleniz> **Ali çay içebildi mi. (Could Ali drink tea?)**
T_agent: Evet :) (Yes)

Figure 6.3. Sample dialogue using the MSF mechanism

The MSF mechanism generates an answer if the agent, theme, and location of a member of the general information list are identical to the corresponding arguments of the search list.

In the DOSSF mechanism, we add a deduction part based on the semantic network hierarchy explained in Section 3.4. Let us say that the agent knows that Kemal drinks tea at breakfast, and the user asks whether Canan can drink tea at breakfast. Kemal and Canan are both human beings and they belong to the same class in the object hierarchy used by the program. So, if one of them performs the action of drinking, why not the other? This dialogue is given in Figure 6.4. This mechanism is based on the identification of such similarities between the theta roles in the lists.

In the DOSSF mechanism, we try to match two lists if the semantic class of their agents, themes, and locations are the same. The tense and the auxiliary tense are not taken

into account. If such a match is found, the agent outputs “Mümkündür” (Possibly) and skips the remaining members in the general information list. If the agent can not find a match at the end, it outputs “Bilmiyorum” (I don’t know). If the deontic modality of the search list is ability, then this mechanism is used, otherwise the agent skips this strategy.

Türkçe Okur Yazar (Literate in Turkish)
 Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)

Cümleniz> **Kemal çay içti. (Kemal drank tea)**
 çay(x(1)).
 id_table(x(1), _G2422^çay(_G2422)).
 iç(_G1946, kemal, x(1), [_G1955, _G1958, _G1961, _G1964], definite_past, none,
 pos, generic).
 Teşekkürler, öğrendim...

Cümleniz> **Canan çay içebilir mi. (Can Canan drink tea?)**
 T_agent: Mümkündür (-) (Possibly)

Cümleniz> **Canan çay içti mi. (Did Canan drink tea?)**
 T_agent: Bilmiyorum :((I don’t know)

Figure 6.4. Sample dialogue using the DOSSF mechanism

6.3. Explanation Process and Sentence Generation

When the agent is in “explanatory mode” and the answer is formed by using either the SSF or the MSF mechanism, it tries to explain why it generated the previous answer. For this purpose, the agent determines whether a Prolog rule or a fact was used in answer generation. If a rule was used, the agent converts each verb fact in the conditional part of the rule to a Turkish natural language sentence and outputs them. If a fact was used, the agent states that the queried information had been taught to it earlier.

The mode of the agent can be toggled between “explanatory mode” and “brief mode” during the execution by entering “Detaylı konuş” (Talk in detail) or “Kısa konuş” (Talk briefly). If the latter one is entered, the agent only outputs the answer, and not the reasons. Initially, the agent is in “brief mode”.

If the agent knows that Canan is beautiful, and the user asks whether Canan is beautiful or not, then the answer is generated by TOY’s standard strategy. If the mode is explanatory, the agent outputs “Evet” (Yes) by using only the fact “güzel(Canan)” and outputs that it already learned this as given in Figure 6.5.

Türkçe Okur Yazar (Literate in Turkish)
Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)

Cümleniz> **Canan güzeldir. (Canan is beautiful)**
güzel(canan).
Teşekkürler, öğrendim...

Cümleniz> **Canan güzel midir. (Is Canan beautiful?)**
T_agent: Evet :) (Yes)

Cümleniz> **detaylı konuş. (Talk in detail)**

Cümleniz> **Canan güzel midir. (Is Canan beautiful?)**
T_agent: Evet :) (Yes)
"Canan güzeldir" daha önce öğretilmişti (“Canan is beautiful” was already taught)

Figure 6.5. Sample dialogue where the mode of the agent changes

If rules in the knowledge base are used in the formation of an answer, all free variables in the facts at the conditional part of the rules are instantiated with values. During the answering of the question “Didem aşçı mıdır?” (Is Didem a cook?) by using the rule and facts given in Figure 6.1, the rule becomes

aşçı(didem):-yemek(x(2)), pişir(_G2202, didem, x(2), [_G2211, _G2214, _G2217, _G2220], _G2206, none, pos, generic), evkadını(didem).

To create the explanation for this question, the agent must convert verb facts, (pişir(_G2202, didem, x(2), [_G2211, _G2214, _G2217, _G2220], _G2206, none, pos, generic) in this case), and the non-verb facts including proper noun constants as arguments, (evkadını(didem) in this case), to sentences.

Non-verb facts to be converted to sentences can have the forms noun(propernoun) (like evkadını(didem)) or adjective(propernoun) (like güzel(didem)). In each case, the proper noun becomes the subject and the noun or the adjective becomes the nominal verb of the sentence. (“Didem evkadınıdır” (Didem is a housewife) or “Didem güzeldir” (Didem is beautiful)).

The procedure which generates sentences from verb facts has two major components; finding the referents of the discourse markers and constructing the sentence structure. In verb facts, either proper nouns or discourse markers (whose referents have to be found) sit in theta roles. We search the id_table to find the corresponding semantic formulas of each discourse marker in the verb facts.

pişir(_G2202, didem, x(2), [_G2211, _G2214, _G2217, _G2220], _G2206, none, pos, generic).

In the above fact, the agent is a proper noun and does not need a conversion, but the theme has to be converted to a noun.

Before going into the details of finding the referent of a discourse marker, let us review some features of the “parse” predicate which is explained in Section 4.3.

The “parse” predicate outputs two lists and the structure of the first output list is

[type of the word, semantic formula of the word, names of the affixed suffixes]

The difference between these lists of words like “yemek” (meal) and “yemekler” (meals), which have the same root and type is the names of the affixed suffixes as shown in Figure 6.6.

```
?- parse([y,e,m,e,k],A_list1,A_list2).

A_list1 = [type(noun), sem(_G288^yemek(_G288)), num(sing)]
A_list2 = [[sm, cns, kt], [ek(6)]] .

?- parse([y,e,m,e,k,l,e,r],A_list1,A_list2).

A_list1 = [type(noun), sem(_G309^yemek(_G309)), num(plur)]
A_list2 = [[sm, cns, kt], [ek(7)]] .
```

Figure 6.6. Parsing of the words “yemek” (meal) and “yemekler” (meals)

By using the reversibility advantage of Prolog programming, if the second argument is given as input to the “parse” predicate and the third argument is left anonymous, the word that is associated with this list can be retrieved, as given in Figure 6.7.

```
?- parse(A, [type(noun), sem(_G288^yemek(_G288)), num(sing)],_).

A = [y,e,m,e,k]
```

Figure 6.7. Retrieving of the word “yemek” (meal) from the “parse” predicate

In order to convert the $x(2)$ in our example to a noun, one has to construct the second argument list associated with the parse of the referent of $x(2)$. The first member of the list is “type(noun)”. The second member, the semantic formula, is retrieved by using the `id_table` entry, `id_table(x(2), _G2828^yemek(_G2828))`. As shown in Figure 6.6, the first two members can be the same for different inflections of the same stem. In order to retrieve the correct one, the agent uses the “words_content” list, which was explained in Section 5.4.4.

After the sentences shown in Figure 6.1 are processed and the facts are asserted to the KB, the words_content list looks like:

```
words_content = [[didem, num(sing), definiteness], [x(2), num(sing)]]
```

Each member of this list is a list that includes the word or its discourse marker and the list that includes the names of the affixed suffixes.

The agent obtains the list that includes the names of the affixed suffixes from this list. In this case, the rest of the list which includes x(2) is [num(sing)]. Then the agent combines it with the type and the semantic formula and uses them as the second argument in the parse predicate. Finally it retrieves the referent of the discourse marker (x(2)) as shown in Figure 6.7.

In the conversion of verb facts to verbs, the words_content list is not used. With the exception of the number of person, the required values like tense, auxiliary tense, etc. can be obtained from the verb fact. The number of person is retrieved from the agent, by checking whether the agent is singular or plural. If the sublist of the words_content list that includes the agent has “num(sing)” as a member, the number argument of the verb is set to singular, otherwise it is set to plural. Again, the first output list is constructed and used in the “parse” predicate to obtain the verb as given in Figure 6.8.

```
?- parse(Word,[type(verb), sem(_G303^_G306^_G309^pişir(_G312, _G309,
_G306, [_G321, _G324, _G345, _G871], _G316, _G317, _G318, _G732)), st2(pos),
tense(aorist), st1(statement), aux tense(none), per(3), num(3)],_).

Word = [p,i,ş,i,r,i,r].
```

Figure 6.8. Retrieving of the verb “pişirir” (foods) from the “parse” predicate

If the tense argument is a free variable as in Figure 6.1, “tense(aorist)” is added to the first output list before the execution of the “parse” predicate.

When each theta role of the verb fact has been retrieved, the final step is the construction of the sentence. The orders used in TOYagent are:

- agent-theme-location-verb: if the verb is transitive and the location information is given
- agent-theme-verb: if the verb is transitive and the location information is not given
- agent-location-verb: if the verb is intransitive

The order used in the generated sentence may be different from the one used in the original sentence that may have caused the insertion of this fact to the KB.

For this case (aşçı(Didem)), the reasons are expressed as shown in Figure 6.9.

<p>Agent: Didem</p> <p>Theme:</p> <p>Loc:</p> <p>Verb: evkadınıdır.</p> <p>Agent: Didem</p> <p>Theme: yemek (x(2))</p> <p>Loc:</p> <p>Verb: pişirir.</p> <p>Sentences:</p> <p>“Didem evkadınıdır” (Didem is a housewife)</p> <p>“Didem yemek pişirir” (Didem cooks a meal)</p>

Figure 6.9. Sentence construction example

In Figure 6.10, it is given that “All little children are cute” and “Ali is a little child”. If the user asks “Is Ali cute?” and the agent is in explanatory mode, the agent outputs “Evet” (Yes). The agent finds the predicates küçük(ali) and çocuk(ali) as part of the

information it used during the formation process. The agent expresses them one by one and does not combine these two predicates. As a result, the predicates are expressed as “Ali küçüktür” (Ali is little) and “Ali çocuktur” (Ali is a child), although the combined expression “Ali küçük bir çocuktur” (Ali is a little child) would be more appropriate.

Sentences which have previously been entered:

Bütün küçük çocuklar sevimlidirler. (All little children are cute)

Ali küçük bir çocuktur. (Ali is a little child)

Predicates in the KB:

sevimli(_G1392):-küçük(_G1392), çocuk(_G1392).

küçük(ali).

çocuk(ali).

Figure 6.10. Content of the knowledge base during a sample execution

Another shortcoming of TOYagent is obtaining the singularity or plurality information and case information (accusative, or nominative) from the words_content list. In TOY, when the semantic representation of a sentence is transformed into facts and asserted to the KB, number and case information is lost (see Section 5.4). The correct solution to this shortcoming is changing the KB structures and putting necessary slots for storing this kind of information that will be used in sentence generation. In TOYagent, in order to solve this inherited problem, we opted to obtain all this information from the words_content list. The words_content list includes information about the facts learned during the conversation session. For facts already in the KB pre-coded before the session, no number and case information is kept, and sentence generation will generate incorrect sentences from these facts.

Ögün [8] implements an explanation mechanism in his program to express the reasons of a yes/no question by Turkish natural language sentences. The explanation is given if the user asks a “Neden?” (Why?) question. The information in “provelist”, which is a global list keeping successful unification steps, is converted to Turkish sentences in

two steps: the conversion of the knowledge base entries into a semantic formula, and the conversion of the semantic formula to a natural language sentence.

At the first step, an intermediate logical formula is created from the entries in the knowledge base. The quantifier definitions corresponding to each theta role in an event entry is retrieved for that purpose. At the second step, a set of predicates, the core of which is the “generatets” predicate, are used for generating the sentence from its logical formula. Details and examples of this mechanism can be seen in [8].

7. CONCLUSION

This chapter gives a summary of our work and a preview of future extensions.

7.1. TOYagent

In this thesis, a conversational agent named TOYagent, whose treatment of word meaning is improved in several respects with respect to its predecessors, is explained. The agent participates in conversations with the user and tries to learn something from the user or answer questions of the user. The processing of the input sentence is determined according to its mood. If it is a statement sentence, the agent tries to extract the knowledge from it, otherwise, it skips to question answering.

When the user enters a sentence to the agent, it is processed through three levels: morphological, syntactic, and semantic. These levels are inherited from the natural language infrastructure developed by Çetinoğlu [1] and required additions and modifications are done on this infrastructure in our work.

A lexicon is used to store the roots and the suffixes which are defined using the same data structure. The lexicon forms the scope of the words that can be used in input sentences. One of the improvements that we made is allowing the program to engage the user in “word-learning” subdialogues to extend the lexicon in an “online” manner.

In the morphological level, each word in the input sentence is morphologically checked. Checking means controlling the existence of the word’s root in the lexicon and controlling whether that word can be constructed by affixation of the suffixes in the lexicon to the root. If the agent recognizes a word that does not have an entry in the lexicon, it asks questions to the user for the determination of some of the slot values in the lexicon entry that it goes on to create for that word. The user also gives a brief description of the meaning of the unknown word in the context of other words. This process brings a higher flexibility to the agent’s execution. We also added a stemming module which is used during this process.

In the syntactic level, the syntactic construction of the sentence is checked. The defined DCG rules are used during this process and at the end the semantic representation of the sentence is obtained. We added new DCG rules to the agent and some of these rules make the agent accept conjoined sentences as input. One of our improvements in this level is the treatment of participle clauses in sentences.

We also added an anaphora resolution mechanism to the agent. Using this mechanism, referents of the personal pronouns in the input sentence are identified and replaced with the pronouns in the semantic representation.

The semantic representation of the sentence indicates the mood of the sentence (either statement or question). In the semantic level, if the mood is statement, the agent extracts the knowledge from the representation and stores it in its knowledge base. Representation of the knowledge in the KB is obtained by applying certain transformation routines on the semantic representation. One of our improvements has been the addition of new routines (e. g., for the transformation of participle clauses) to this transformation module.

We also enabled the agent to use a semantic network hierarchy which provides commonsense information about the external world. Each word in the lexicon is classified according to this hierarchy. This network is used in the anaphora resolution and answer formation mechanisms.

If the user asks a yes/no question to the agent, it tries to form an answer. Three mechanisms are used in answer formation, one of which is inherited from TOY. One of these mechanisms uses deontic modalities and another makes deductions based on the semantic network hierarchy. The two additional mechanisms that we implemented improve the answering capability of the agent, especially in uncertain conditions.

One of our improvements is a module for sentence generation from the knowledge in the KB. Using this module, the agent can explain the reasons of the formed answer by Turkish natural language sentences.

7.2. Future Work

In this section, we suggest some solutions to several of TOYagent's present shortcomings:

- In TOYagent, morphological and syntactic controls are performed during the construction of the semantic representation of the sentence. However, no “commonsense check” is present. So if a sentence like “Dağlar gökyüzünde uçuyordu” (The mountains were flying in the sky) is given to the agent, its semantic representation can be produced without any objection. The semantic network hierarchy can be used for implementing a commonsense control module.
- In natural dialogues, sentences can have missing elements which can be filled in by the hearer by using the preceding sentences in the discourse. (This process is called ellipsis [18]) In TOYagent, there is no mechanism that can find the referents of the missing elements and place them in the semantic representation of the sentence; so the semantic representation of such a sentence can not be obtained in the agent. In the future, the necessary mechanisms for ellipsis can be added to the agent, and again the semantic network can be used in these mechanisms.
- If the agent learns something from the user, there is no way of removing this knowledge from the KB during the dialogue, even if the user explicitly declares that what he said one sentence ago was incorrect. A more advanced mechanism for the management of the KB can be added to the agent [8].
- Only one user can talk with the agent. The number of users can be increased and the information given by a user can be passed to others [8].
- Possessive suffixes are not handled in the agent. If an object that belongs to somebody is used in a sentence, there is no way to link the object to the owner in the semantic representation. Required changes in the semantic representation for these suffixes can be implemented [8].
- The lexical preprocessor is presently used in the addition of new entries for unknown words. But with the necessary modifications on the processor, it can also be used for correcting the words in the input sentence. If the sentence “Bütün anneler lezzetli yemeklar pişirirler” (All mothers cook delicious meals) is given to the preprocessor, it can recognize that the word “yemeklar” (*meals) is

morphologically incorrect, and ask the user to correct this entry before the sentence processing.

- The anaphora resolution mechanism can be improved by taking the whole conversation into account. The agent currently accepts the input sentences and tries to find the referent of an anaphor, if any, only in these sentences. So, if the sentence “O öğrencidir ve o başarılı bir insandır” (he/she is a student and he/she is a successful person)) is given to the agent, it can not find the referents of the anaphors. If the whole conversation is taken into account, the referent can be found. Cataphora resolution can also be added to the agent. The improved mechanism can also find the subject if it is not used in the sentence as in the sentence “okula gitmediler” (*They* didn’t go to the school).
- The agent currently answers only yes/no questions. With the integration of the wh-question modules of TOY with the new lexicon and DCG design, it can be made to answer any type of question in Turkish.
- The semantic network hierarchy can be improved so that a more general classification can be retrieved.
- A better integration of the semantic network hierarchy and the knowledge base can be achieved. If the KB has explicit facts about concepts that also appear in the semantic hierarchy, (like “human” and “animate_object”) the program should be able to make use of the subset relationship among these stated in the network without the need for explicit addition of the rule

animate_object(X):-human(X).

by the programmer.

- The morpheme structure of each verb must be modified and some necessary slots have to be added to the agent in order to solve the problems in sentence generation from the predicates in KB.
- In TOYagent, when a sentence like “Ali araba kullanabilir” is processed, the meaning of the sentence is accepted as “Ali can drive a car“ and it is represented by the semantic formula
`singsome(_G1547, araba(_G1547), kullan(_G2009, ali, _G1547, [_G2018, _G2021, _G2024, _G2027], _G6432, none, pos, abil)).`

The correct meaning of the sentence is “Ali can drive any car” and it must be represented by the formula

`plurall(_G1661, araba(_G1661) kullan(_G2483, ali, _G1661, [_G2492, _G2495, _G2498, _G2501], _G8792, none, pos, generic)))`.

Necessary modifications must be done for solving this problem.

APPENDIX A: AN EXAMPLE DIALOGUE WITH THE AGENT

Türkçe Okur Yazar (Literate in Turkish)

Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)

Cümleniz> **bütün anneler güzeldirler. (All mothers are beautiful)**

T_agent: anneler kelimesini bilmiyorum (I don't know the meaning of the word "anneler")

Kelimenin Kökü hangisi: (Which one is the root of the word?)

1 : anne

2 : anneler

Hangisi (Which one): **1.**

T_agent: anne kelimesinin türü nedir ? (What is the type of the word "anne")

1:İsim (Noun)

2:Sıfat (Adjective)

3:Zaman (Time_unit)

4:Özel İsim (Propernoun)

Hangisi (Which one):**1.**

T_agent: "anne kelimesinin tanımıyla ilgili sorular sorayım mı?(e/h) (Shall I ask questions about the word's description to you?)

! : **e.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:

somut soyut

> **somut. (concrete)**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:

canlı cansız

> **canlı. (animate)**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:

insan hayvan bitki

> **insan. (human being)**

tr_morph_entry(AdKök, [[a, n, n, e], [type(noun), sem(_G1512^anne(_G1512))], [sm,cn,i]], _, _, _, e, e, ok)

T_agent:anne ne demek? Anlatır mısın? (Can you explain what a mother is?)

l: **çocuğu olan bir bayan annedir ve her anne melektir. (A woman who has a child is a mother and every mother is an angel)**

T_agent: melektir kelimesini bilmiyorum (I don't know the meaning of the word "melektir")

Kelimenin Kökü hangisi: (Which one is the root of the word?)

1 : melek

2 : melektir

3 : meleş

Hangisi (Which one): **1.**

T_agent: melek kelimesinin türü nedir ? (What is the type of the word "melek")

1:Isim (Noun)

2:Sıfat (Adjective)

3:Zaman (Time_unit)

4:Özel Isim(Propernoun)

Hangisi (Which one):**1.**

T_agent: "melek kelimesinin tanımıyla ilgili sorular sorayım mı?(e/h) (Shall I ask questions about the word's description to you?)

l: **e.**

T_agent:Aşağıdaki sınıflardan hangisine giriyor:

somut soyut

>soyut. (abstract)

tr_morph_entry(AdKök, [[m, e, l, e, ğ], [type(noun), sem(_G5803^meleş(_G5803))], [sy]], _, _, _, e, ğ, spec)

[m, e, l, k] kelimesi doğru mu? e/h **h.** (Is the word "melk" correct?)

tr_morph_entry(AdKök, [[m, e, l, e, k], [type(noun), sem(_G5656^melek(_G5656))], [sy]], _, _, _, e, k, specok)

T_agent: melek ne demek? Anlatır mısın? (Can you explain what an angel is?)

! melek kutsaldır. (An angel is a holy thing)

semantic formula: singthe(_G6780, melek(_G6780), kutsal(_G6780))

kutsal(_G6780):-melek(_G6780)

Teşekkürler, öğrendim...

semantic formula:singsome(_G7812, (singsome(_G7837, çocuk(_G7837), var(_G8618, _G7812, _G7837, [_G8627, _G8630, _G8633, _G8636], _G8622, _G8623, _G8624, _G5443)), bayan(_G7812)), anne(_G7812)), singall(_G10176, anne(_G10176), melek(_G10176))

anne(_G7812):-çocuk(_G7837), var(_G8618, _G7812, _G7837, [_G8627, _G8630, _G8633, _G8636], _G8622, _G8623, _G8624, _G5433), bayan(_G7812)

semantic formula: singall(_G10176, anne(_G10176), melek(_G10176))

melek(_G10176):-anne(_G10176)

Teşekkürler, öğrendim...

semantic formula:plurall(_G11970, anne(_G11970), güzel(_G11970))

güzel(_G11970):-anne(_G11970)

Teşekkürler, öğrendim...

Cümleniz> **Canan anne midir. (Is Canan a mother?)**

Bilmiyorum :((I don't know)

Cümleniz> **Canan melek midir. (Is Canan an angel?)**

Bilmiyorum :((I don't know)

Cümleniz> Canan çocuđu olan bir bayandır.

semantic formula: singsome(_G2410, çocuk(_G2410), var(_G3195, canan, _G2410, [_G3204, _G3207, _G3210, _G3213], _G3199, _G3200, _G3201, _G3202)), bayan(canan)

çocuk(x(1))

id_table(x(1), _G3991^çocuk(_G3991))

var(_G3195, canan, x(1), [_G3204, _G3207, _G3210, _G3213], _G3199, _G3200, _G3201, _G3202)

bayan(canan)

T_agent: Teşekkürler, öğrendim...

Cümleniz> **Canan anne midir. (Is Canan a mother?)**

Evet :) (Yes)

Cümleniz> **Canan güzel midir. (Is Canan beautiful?)**

Evet :) (Yes)

Cümleniz> **Canan melek midir. (Is Canan an angel?)**

Evet :) (Yes)

APPENDIX B: SYSTEM INTERFACE

By entering *Toy*, the user starts the conversation with the agent and he/she ends it by entering *hoşçakal*. An example follows:

```

:? Toy.
Türkçe Okur Yazar (Literate in Turkish)
Çıkmak için cümle yerine `hoşçakal` yazın. (Write hoşçakal to end the conversation)

Cümleniz> Ali çay içti. (Ali drank tea)
.....
Cümleniz> hoşçakal. (good bye)

```

The Turkish sentences given by the user must end with a period and the proper nouns must start with a capital letter.

The main source files and their corresponding roles follow:

- arceson.pl: This file contains the declarations of the FSMs used (the initial node, the final nodes and arcs of the FSMs).
- formula.pl: This file contains the main predicates for the transformation and assertion of the logical formula to the knowledge base.
- misc.pl: This file contains the auxiliary functions used.
- morphoson.pl: This file contains the morpheme lexical entries, including root, suffix and number entries. The procedures for morphological processing are also contained here.
- sohbetson.pl: This file contains the main procedures for syntactic processing. It contains a set of DCG rules designed in the scope of the thesis.
- semnet.pl: This file contains the declarations of the nodes in the semantic network hierarchy.
- functions.pl: This file contains the predicates for solving the anaphora resolution problem.

- functions2.pl: This file contains the predicates for asserting the knowledge in conjoined sentences.
- functions3.pl: This file contains the predicates for learning new words and retrieving their descriptions from the user.
- muhabbet.pl: This file contains the main predicates for processing the sentences and for answering the questions given by the user.
- answer.pl: This file contains the predicates for finding the reasons of the formed answer and constructing sentences from facts in the knowledge base.

APPENDIX C: IMPLEMENTATION ENVIRONMENT

The program is encoded by using SWI-Prolog version 5, which is a non-commercial academic program developed in University of Amsterdam. The program can run both on Unix and on Windows. The setup programs for each can be downloaded from [26].

SWI-Prolog has no specific system requirements. It can run on a PC that uses one of the Unix, Windows95/98/NT/2000/XP/ME operating systems. The program is developed on a PC using Windows ME as the operating system.

REFERENCES

1. Çetinoğlu, Ö., 2001, *A Prolog Based Natural Language Infrastructure for Turkish*, M.S. Thesis, Boğaziçi University.
2. Garigliano, R., A. Urbanowicz and D. J. Nettleton, 1998, "Description of the LOLITA System as Used in MUC-7", *Proceedings of the MUC-7*, Washington, 7-9 April 1998 .
3. Smith, M. H., R. Garigliano and R. G. Morgan, 1994, "Generation in the LOLITA System: An Engineering Approach.", *7th International Generation Workshop*, Kennebunkport, Maine, 21-24 June 1994.
4. A Conversational System Architecture:Galaxy, <http://www.sls.lcs.mit.edu/sls/whatwedo/architecture.html>, 2003.
5. Allen, J., D. Byron, M. Dzikovska, G. Ferguson, L. Galescu and A. Stent, 2001, "Towards Conversational Human-computer Interaction", *AI Magazine* 22(4), pp. 27-38.
6. Allen, J., G. Ferguson, and A. Stent, 2001, "An Architecture for More Realistic Conversational Systems", *Proceedings of Intelligent User Interfaces 2001 (IUI-01)*, Santa Fe, NM, 14-17 January, 2001.
7. Searle, J., 1969, *Speech Acts: An Essay in the Philosophy of Language*, Cambridge University Press, Cambridge.
8. Ögün, F., 2003, *Design and Implementation of an Improved Conversational Agent Infrastructure for Turkish*, M.S. Thesis, Boğaziçi University.
9. Şeker, Ş.E., 2003, *Turkish Speaking Assistant (TuSA)*, M.S. Thesis, Yeditepe University.

10. Russell, S. and P. Norvig, 2003, *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey.
11. Cycorp Web Page, <http://www.cyc.com>, 2003.
12. Mayfield, J., T. Finin, R. Narayanaswamy, C. S. W. MacCartney and K. Goolsbey, 1995, "The Cycic Friends Network: Getting Cyc Agents to Reason Together", *Proceedings of the CIKM '95 Workshop on Intelligent Information Agents*, December 1995.
13. Princeton WordNet Web Page, <http://www.cogsci.princeton.edu/~wn/index.shtml>, 2003.
14. Euro WordNet Web Page, <http://www.illc.uva.nl/EuroWordNet/index.html>, 2003.
15. Balkanet Web Page, <http://www.ceid.upatras.gr/Balkanet>, 2003.
16. Turkish WordNet Web Page, <http://people.sabanciuniv.edu/~oflazer/balkanet/index.htm>, 2003.
17. McClelland, J.L., D.E. Rumelhart and the PDP Research group, 1998, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition-II Psychological and Biological Models*, MIT press, Cambridge.
18. Covington, M.A., 1994, *Natural Language Processing for Prolog Programmers*, Prentice Hall, New Jersey.
19. Carbonell, J. and R. Brown, 1998, "Anaphora Resolution: A Multi-strategy Approach", *Proceedings of 12th International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August 1998.
20. Tin, E. and V. Akman , 1994, "Situated Processing of Pronominal Anaphora", *Proceedings of the KONVENS'94 Conference*, Vienna, 28-30 September 1994.

21. Ge, N., J. Hale and E. Charniak, 1998, "A Statistical Approach to Anaphora Resolution", *Proceedings of the Sixth Workshop on Very Large Corpora*, Montr'eal, Canada, August 1998.
22. Kennedy, C. and B. Boguraev, 1996, "Anaphora for Everyone: Pronominal Anaphora Resolution Without a Parser", *Proceedings of 16th International Conference on Computational Linguistic (COLING '96)*, 5-9 August 1996, pp 113-118.
23. Mitkov, R., L. Belguith and M. Stys, 1998, "Multilingual Robust Anaphora Resolution", *Proceedings of the Third International Conference on Empirical Methods in Natural Language Processing (EMNLP-3)*, Granada, Spain, June 1998, pp. 7-16.
24. Olsen, M. B., B. J. Dorr and S. C. Thomas, 1998, "Enhancing Automatic Acquisition of Thematic Structure in a Large-Scale Lexicon for Mandarin Chinese", *Proceedings of the Third Conference of the Association for Machine Translation in the Americas, AMTA-98*, Langhorne, PA, 28-31 October 1998, pp. 41-50.
25. Oflazer, K., 1993, "Two-level Description of Turkish Morphology", *The Second Turkish Symposium on Artificial Intelligence and Neural Networks*, Ankara, June 1993, pp 86- 93.
26. SWI- Prolog Web Page, <http://www.swi.psy.uva.nl/projects/SWI-Prolog/>, 2003.

REFERENCES NOT CITED

Özsoy, A. S., *Türkçe = Turkish*, Boğaziçi Üniversitesi Yayınları, İstanbul, 1999.