DESIGN AND IMPLEMENTATION OF A SYSTEM FOR MAPPING TEXT MEANING REPRESENTATIONS TO F-STRUCTURES OF TURKISH SENTENCES

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER

ENGINEERING AND INFORMATION SCIENCE

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Selman Murat Temizsoy

August, 1997

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

Asst. Prof. İlyas Çiçekli (Principal Advisor)

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

Assoc. Prof. Halil Altay Güvenir

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

Asst. Prof. Özgür Ulusoy

Approved for the Institute of Engineering and Science:

Prof. Dr. Mehmet Baray
Director of Institute of Engineering and Science

ABSTRACT

DESIGN AND IMPLEMENTATION OF A SYSTEM FOR MAPPING TEXT MEANING REPRESENTATIONS TO F-STRUCTURES OF TURKISH SENTENCES

Selman Murat Temizsoy
M.S. in Computer Engineering and Information Science
Advisor: Asst. Prof. İlyas Çiçekli
August, 1997

Interlingua approach to Machine Translation (MT) aims to achieve the translation task in two independent steps. First, the meanings of source language sentences are represented in a language-independent artificial language. Then, sentences of the target language are generated from those meaning representations. Generation task in this approach is performed in three major steps among which the second step creates the syntactic structure of a sentence from its meaning representation and selects the words to be used in that sentence. This thesis focuses on the design and the implementation of a prototype system that performs this second task. The meaning representation used in this work utilizes a hierarchical world representation, *ontology*, to denote events and entities, and embeds semantic and pragmatic issues with special frames. The developed system is language-independent and it takes information about the target language from three knowledge resources: *lexicon* (word knowledge), *map-rules* (the relation between the meaning representation and the syntactic structure), and target language's *syntactic structure representation*. It performs two major tasks in processing the meaning representation: lexical selection and mapping the two representations of a sentence. The implemented system is tested on Turkish using small-sized knowledge resources developed for Turkish. The output of the system can be fed as input to a tactical generator, which is developed for Turkish, to produce the final Turkish sentences.

Keywords: Machine Translation, Interlingua Approach, Natural Language Generation, Text Meaning Representation, Syntactic Structure Representation, Ontology, Lexicon

ÖZET

METİN ANLAMSAL GÖSTERİMLERİNİN TÜRKÇE CÜMLE YAPILARINA DÖNÜŞTÜREN BİR SİSTEMİN TASARIMI VE UYGULAMASI

Selman Murat Temizsoy
Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans
Danışman: Yrd. Doç. Dr. İlyas Çiçekli
Ağustos, 1997

Bilgisayarla Çeviri problemine *Interlingua* yaklaşımı çeviri sorununu birbirinden bağımsız iki aşamada gerçekleştirmeyi amaçlar. Önce, kaynak dildeki cümlelerin anlamları doğal dilden bağımsız, yapay bir dilde temsil edilir. Sonra, hedef dildeki cümleler bu anlamsal gösterimlerden üretilir. Metin üretim görevi bu yaklaşımda üç ana aşamada gerçekleştirilir ve ikinci basamakta anlamsal gösterimden cümlenin yapısal özellikleri çıkartılır ve cümlede kullanılacak sözcükler seçilir. Bu tezde bu ikinci basamağı gerçekleştirebilecek prototip bir sistemin tasarımı ve uygulaması amaçlanmaktadır. Bu çalışmada kullanılan anlamsal gösterim olayları ve varlıkları temsil edebilmek için dünyanın sıradüzensel bir gösterimi olan *ontoloji*den yararlanmaktadır ve ayrıca bu gösterim anlamsal ve pragmatik özellikler için farklı yapılar kullanmaktadır. Geliştirilen sistem dilden bağımsızdır ve dile ait bilgileri üç ayrı bilgi kaynağından alır: *sözlük* (anlamsal ve yapısal sözcük bilgisi), *dönüştürme-kuralları* (anlamsal gösterimle cümle yapıları arasındaki bağlantı), ve hedef dilin *yapısal özelliklerinin gösterimi*. Sistem anlamsal gösterimi işlerken iki ana görevi yerine getirir: sözcük seçimi ve cümlenin iki gösterimi arasında dönüşümü. Uygulanan sistem Türkçe için geliştirilmiş küçük-ölçekli bilgi kaynaklarıyla test edildi. Bu sistemin çıktısı Türkçe için geliştirilmiş bir yüzeysel üreticinin yardımıyla amaçlanan Türkçe cümlelerin üretilmesinde kullanılabilir.

Anahtar Sözcükler: Bilgisayarla Çeviri, Interlingua Yaklaşımı, Doğal Dil Üretimi, Metin Anlamsal Gösterimi, Sözdizim Yapısal Gösterimi, Ontoloji, Sözlük

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my supervisor Asst. Prof. İlyas Çiçekli for his guidance, suggestions and valuable encouragement throughout the development of this thesis.

I would like to thank Assoc. Prof. Halil Altay Güvenir and Asst. Prof. Özgür Ulusoy for reading and commenting on the thesis and for the honor they gave me by presiding the jury.

I thank my family and my friends Alper, Ayşin, Ebru, Erdem, Evrim, Eylem, and Gürhan, for everything.

Contents

List of Figures

Chapter 1

Introduction

Machine translation (MT), one of the most complex and comprehensive branches of computational linguistics and artificial intelligence, aims at developing systems that take a text in one language, *source language*, and produce a text in another language, *target language*, such that the meaning resides in the source text is transfered into the target text through using knowledge about those languages [12, 13]. So, the black-box model of a machine translation system is defined as the system shown in Figure 1.1.

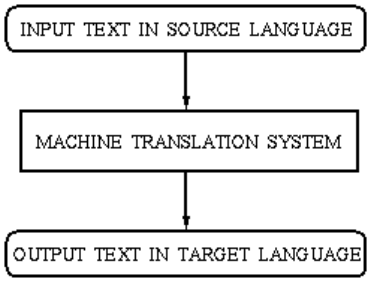

Figure 1.1: Black-Box Model of a Machine Translation System

There are three major computational approaches to machine translation problem: *direct*, *transfer*, and *interlingua* [13, 10]. *Direct* approach carries out the translation task using a large set of language-pair dependent rules for structural and lexical choices. In this approach, there is not any intermediate representation of neither the source nor the target language, and the analysis of the source text directly produces the target text. This approach can be characterized as *word-to-word* translation with some local word-order adjustment. Examples of such systems are SYSTRAN [30] and older versions of SPANAM [29].

Transfer approach, unlike the direct approach, is based on the independent

analysis of the source text from the generation of the target text. Transfer-based MT systems generally produce a kind of syntactic representation of the source text in this analysis phase. Then this representation is translated into the intermediate representation of the target text from which the final target text is generated. So, in this approach, the source and the target language are in direct contact in the translation step between the intermediate representations. This methodology is frequently used for bilingual translation systems since the translation between the two intermediate representations must be developed for every language pair in a multilingual environment (exponential growth with the increase in the number of languages). Among the transfer based translation systems are EUROTRA [1] and METAL [4].

Interlingua approach, similar to transfer approach, is based on the independent analysis of the source text. The difference of this approach comes from its treatment of the translation step. In interlingua MT systems, the source and the target language are never in direct contact. Instead, a language neutral, artificial meaning representation is produced in the analysis step. This meaning representation is input to the generation phase of the target text. This approach has two major advantages over transfer approach: it is more appropriate for developing multilingual MT systems since the analysis and the generation modules of a language are developed for once, and transfer step is not constrained to neither the source nor the target language because of language-independent representation. But, it has general disadvantages: designing a language-independent representation which covers most of language phenomena is difficult, and both the analysis and the generation phases become more complicated. This approach stresses the fact that meaning is language-independent, and languages are encoding systems used by humans to present their view of world to each other. Among the systems conforming to the interlingua design are Ultra [8], Kant [26, 21], and Microcosmos [3, 18].

The methodology that is utilized in this work is the interlingua approach [10, 22, 26, 23]. It separates the analysis task from the generation task using an artificial meaning representation. Generally, the analysis step firstly extracts the syntactic structures of the source text sentences, and then produces the meaning representation through a semantic analysis. The generation phase performs these two steps in reverse order, producing the syntactic structures of the target text sentences using the semantic information, and generating the final target sentences from these syntactic structures. This division of the analysis and the generation tasks into two independent steps is based on the observation that

meaning takes certain forms in any natural language. The computational model utilized by interlingua approach is shown in Figure 1.2.

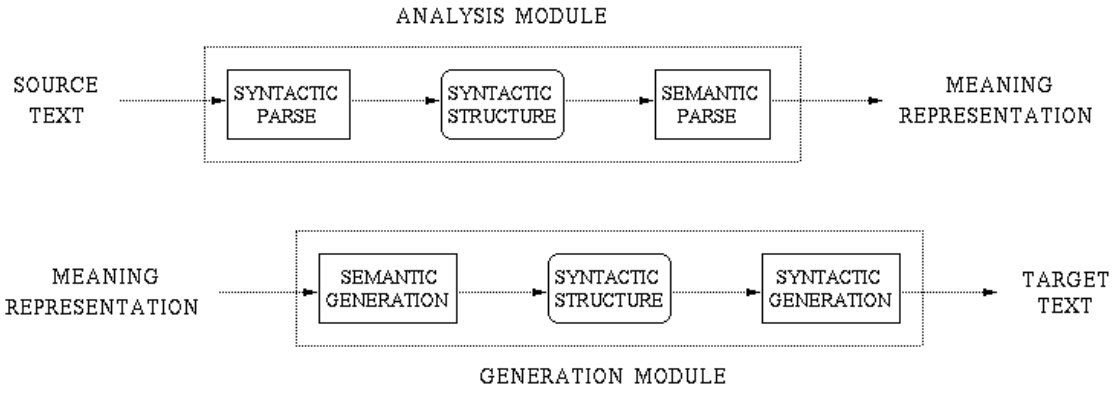

Figure 1.2: Computational Model of Interlingua Systems

The generation step, mentioned above, should perform seven different tasks [22]. *Content delimination* is the phase in which the propositional and the rheoterical goals which are overtly realized in the source text and the remaining goals to be inferred by the text consumer are planned. Determination of the sentences' boundaries of the planned goals is done in *text structuring* phase. Referring to entities without explicitly mentioning them is a common phenomena in languages and text consumer is responsible for making inferences about those entities. *Coreference treatment* phase introduces reference phenomena whenever its usage is appropriate or needed. Open-class lexical items of the target language which are to be used in the target text are selected in *lexical selection* phase. *Syntactic construction* phase is responsible for creating the syntactic structure of each planned sentence from its meaning representation, and introducing closed-class lexemes to the target text whenever needed. Determination of the word ordering of a sentence, which is also a common phenomenon in languages, is achieved in *constituent ordering* phase. The final phase, *realization*, introduces necessary morphological markings to the words and produces the final sentences. These seven tasks defined above can be grouped into three major phases in generation task [22]:

1. *Text Planning*: Performs the first two tasks, *content delimination* and *text structuring*, and returns the meaning representation of every individual sentence to be appeared in the target text.

2. *F-Structure Creation*: Performs the next three tasks, *coreference treatment*, *lexical selection*, and *syntactic construction*, and returns the complete

syntactic structure of each sentence with lexical items inserted.

3. *Tactical Generation*: Performs the last two tasks, *constituent ordering* and *realization*, and generates the final target sentences.

The goal of this work is to design a prototype system that performs the second task, *f-structure creation*, in a language independent way. The developed system takes the meaning representation of a sentence as input and constructs the syntactic structure of the target sentence as output by utilizing various knowledge resources fed into the system. In other words, the system makes transfer between two representation languages, the *text meaning representation*, a frame-based, artificial language for representing the propositional content of a sentence with semantic and pragmatic information embedded, and the *feature structure representation*, also a frame-based, artificial language for representing the syntactic properties of a sentence such as its verbal phrase, its grammatical roles (*subject*, *direct object*, etc.), and its noun phrases [10, 22].

To achieve this task, three knowledge resources are utilized by the system: *ontology*, *lexicon*, and *map-rules* [10, 22]. *Ontology* is a kind of hierarchical world modeling in which the semantic properties of entities and events of the real world are represented in an abstract way. Ontology provides abstract concepts that are used to define propositions in text meaning representation. *Lexicon* provides the morphological, syntactic, semantic, and pragmatic properties of the target language's words. The relationship between the information provided in text meaning representation and the feature structure representation of the target sentence is defined in *map-rules*. The computational architecture of the system designed in this work is described in Figure 1.3.

Note that, there is not any language-dependent information in the developed system. All information about the target language is provided in the lexicon and the map-rules knowledge resources. Currently, the implemented tool is tested on Turkish and the feature structure representation of Turkish is taken from Hakkani [11] in which a tactical generator for Turkish is designed and implemented. The meaning representation utilized in this thesis is taken from the Microcosmos project [18, 3].

Before analyzing the computational model, the necessary linguistic background about semantic and pragmatic phenomena that are covered by the text meaning representation is given in Chapter 2. Then, the structures of the representation languages (text meaning representation and feature structure representation), and the information content of the knowledge-bases (ontology,

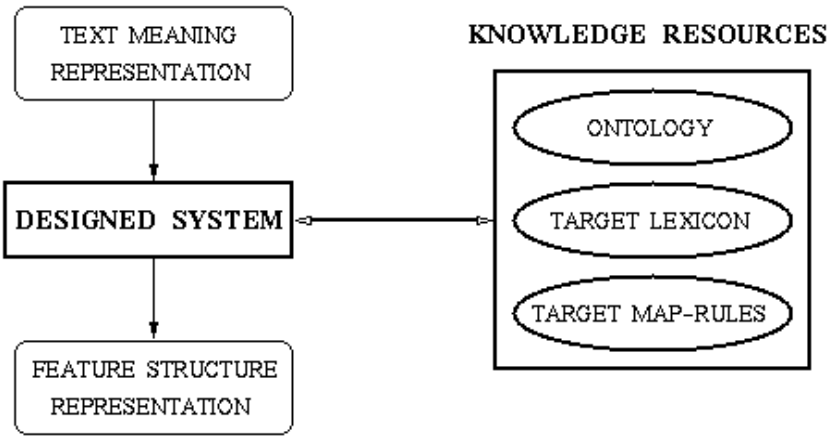

Figure 1.3: Architecture of the Designed System

lexicon, and map-rules) are presented in Chapter 3. Next, the computational model, which makes transfer between the two representation languages, is explained in detail in Chapter 4. Chapter 5 presents the implementation of the described model in Prolog. Finally, the conclusions about this work and future work that can be carried out are given in Chapter 6.

Chapter 2

Linguistic Background

Knowledge-based approach to machine translation, which is the methodology used in this work, is heavily based on the meaning resides in expressions. Translation task in this method is achieved through extracting the functionally complete meaning of a source expression, in which all kinds of ambiguities are removed, and constructing the target expression from this meaning representation. To represent the meaning of an expression, knowledge-based approach utilizes theories from two linguistic fields: *semantics* [9], study of literal meaning that is grammaticalized or encoded, and *pragmatics*, study of meaning that depends on the situation in which an expression is produced.

Semantics deals with the propositional meaning of an expression that can be determinable without any information about the speech context. In other words, it is the study of decontextualized meaning that resides in expressions. The propositional meaning is comprehended by a consumer through matching the producer's model of world with the model of world that is encoded by the expression itself. Languages encode the world with a major distinction between *entities*, independent individuals that are not obliged to be temporarily situated like a human, and *events*, the relations between entities that are essentially tied to change in time like the act of break. Entities are generally encoded as *nouns* and events as *verbs* by languages. Since events are temporal relations between entities, they are represented as predicates that take entities as their arguments with its temporal properties embedded. The set of arguments of an event is limited, and the semantic relations that define the connection between an event and its participants are called as *thematic roles*. The temporal properties of an event are analyzed in two distinct topics: *aspect*, internal structure of an event, and *tense*, temporal relations of an event with other events. The producer's thought about the truth of the expression, its commitment, etc., also affects the

literal meaning and encoded as *modality* in languages.

Pragmatics, in contrast, deals with the contextualized meaning of an expression such as the producer's intention, the consumer's expected response, the situation in which the expression is produced, the historical background, etc. Utterance of an expression causes some kinds of acts to be performed by both the speaker and the hearer, and these acts are explored in pragmatics under *speech-act* topic. *Speech-act* concerns, especially, how the intention of a speaker, like assertion, command, promise, etc., is conveyed by grammatical constructions. Qualification of an expression's component with respect to its relevance, importance, etc., in the communication context is also syntactically realized in languages by word choices, word ordering, etc., and this phenomena is studied in *attitude*. The relationship between the speaker and the hearers, and the social and the cultural context in which communication takes place have an effect on the way an expression is constructed and these issues are analyzed in *stylistics* topic.

Before going into how meaning representation is achieved in knowledge-based approach, the types of semantic and pragmatic information utilized in this representation, *thematic roles*, *aspect*, *tense*, *modality*, *speech-acts*, *attitude*, and *stylistics*, are needed to be explained in detail and the following sections describe each phenomena independently with some demonstrative examples.

2.1 Thematic Roles

Thematic roles can be basically defined as semantic relations that connect entities to events. But this simple definition can cause thematic roles to be confused with other linguistic phenomenon, so this definition should be clarified. First, since events are temporarily situated relations between entities, thematic roles cannot be used for expressions that denote properties of entities, like in "The ball is red". Second, they are not the semantic counterparts of grammatical roles such as *subject*, *direct object*, etc. Grammatical roles are syntactic features of a sentence that can determine the word order, case marking, etc. The distinction can be observed in "It rained ice in Chicago" in which 'it' is the subject of the sentence, but the entity 'it' denotes, weather, clouds, etc., does not participate in the predication and is not associated with any thematic role. Also in passive construction, the grammatical roles of entities are changed, but their thematic roles are remained unchanged (passive construction does not affect the propositional meaning). Third, thematic roles cannot be directly read from

morphological cases. This independence can be exemplified with "*I* have that book" in which '*I*' is marked with *nominative* case in English, but it is marked with *locative* in the Turkish sentence "O kitap *ben-de*" with the same meaning. So, thematic roles must be found in the outside of the systems of morphological cases and grammatical roles, they are constant semantic relationships of predicates and its arguments [9].

Thematic roles are classified into two broad categories: *participant roles*, the arguments necessitated by the predication, and *non-participant roles*, the arguments necessitated by semantic context. Non-participant roles can be extracted from an expression without spoiling the main propositional meaning and they are used to provide contextual information about an event. For example, in sentence "Tom hit the ball in the stadium", 'stadium' can be successfully extracted without disturbing the propositional meaning although deletion of 'ball' results in a meaningless expression. The participant roles are also classified into three categories: *logical actors* (*agent*, *author*, and *instrument*), *logical recipients* (*patient*, *experiencer*, and *benefactive*), and *spatial roles* (*theme*, *source*, *goal*). There are six types of non-participant roles, which are *location*, *path*, *time*, *manner*, *reason*, and *purpose* [9].

2.1.1 Agent

Agent identifies the argument which is the deliberate, potent, or active instigator of a predicate. Agency is generally connected with volition, will, intentionality, and reasonability. So, in sentence "Tommy drove the car", 'Tommy' stands for the agent since he carried out the action deliberately. Even in a situation where he is forced to drive, like in "Terrorists forced Tommy to drive the car", he is still the agent since agency is concerned with the execution, not with the circumstances that give rise to the predicate.

2.1.2 Author

Author, like agent, is the primary executor of a predicate and has all the characteristics of an agent except it is not the direct cause of the act. Author lacks the properties of animacy like volition, intentionality, reasonability, etc. The distinction between the roles agent and author can be shown by sentences "Bill floated down the river" and "The canoe floated down the river". In the first sentence, 'Bill' is the agent because of the deliberateness in the act (if he is unaware of the situation, then this meaning is paraphrased like "Bill's body

floated down the river"). In the second sentence, 'canoe' is the author since it does not carry out the act deliberately.

2.1.3 Instrument

The argument which is the means by which a predicate is carried out is the instrument. Instruments must be acted upon by something else, since they got no energy to carry out an event by themselves. The 'knife' in sentence "Ellen cut the salami with a knife" is the instrument (note that 'Ellen' is the agent). Instruments can be also abstract entities like 'improbable ideas' in "The administration dazzled us with improbable ideas". Note that, even in the absence of an agent, an entity, whose source of energy is external, is marked as an instrument like 'rock' in "The rock broke the window".

2.1.4 Patient

Patient identifies the argument which undergoes, is changed by, or is directly affected by a predicate. Just as the agent is the primary executor of an event, so the patient is the primary recipient. So, 'car' in "The man cleaned the car" and 'glass' in "The boy broke the glass" are the patients of the predicates. Note that, a patient must come out as changed as a result of an action, so 'letter' in "I received a letter" is not the patient of the predicate (it is the theme of the predicate).

2.1.5 Experiencer

Experiencer identifies the argument whose internal state or constitution is affected by a predicate. For example, in "Buddy smelled the flower", if the interpretation of the sentence is such that smell of the flower came over Buddy (does nothing volitionally), Buddy is marked as experiencer (other interpretation is that Buddy smelled the flower volitionally, agent). Since the argument should have an internal state to register the effect, experiencers are generally humans, at least animates. Experiencer generally denotes participant humans who perceive and interpret external data (have a working disposition), take in the data uncontrollably (lack volition), or respond subjectively (have private worlds).

2.1.6 Benefactive

Benefactive identifies the argument that derives actions or entities from the actions of others in predicates. For example, in "Dr. Frankenstein made his son a monster", if the interpretation of the sentence is such that 'son' comes to the possession of a monster, then it is marked as benefactive (other interpretation is that Dr. Frankenstein converted his son into a monster, patient). Note that, neither the goodness of the result (in "Tom lost the game for his team",'team' is the benefactive), nor the co-optation of the constituent (in "Mary bought lunch for Bob", 'Bob' is the benefactive) is required for marking an argument as benefactive.

2.1.7 Theme

Theme identifies the argument that denotes the displaced entity in a motion event like 'arrow' in "Tom shot the arrow through the air". Although there is a similarity between the roles patient and theme (both undergoes acts), themes are different in that they are not modified by the displacement itself. Note that, 'letter' in "I received a letter" is the theme of the predicate since 'letter' denotes the argument that is the displaced entity in the predicate.

2.1.8 Source

Source identifies the argument that denotes the point of origin in motion events. So, 'Ireland' in "Bob was flown in from Ireland" is the source of the predicate. Sources, as the points of origins of predications, are not purely restricted to spatial events, they can be found in events that express any actional or stative sources, like 'sun' in "The sun gives off heat" and 'wine' in "Wine can turn into a vinegar". Note that, 'heat' in the first sentence is the theme and 'vinegar' in the second one is the goal (explained in the next section).

2.1.9 Goal

Goal identifies the argument that denotes the destination point of motion events. So, 'England' in "My wife went to England" is the goal of the predicate. Like sources, goals can denote entities in events that express any actional or stative destinations, like 'Ellen' in "I told Ellen a story". The same observation made in the analysis of sources is valid, abstract entities, like 'story' in the previous

example sentence, can be themes of predicates which have destination arguments. So, in "His thoughts run from liberian to Libertarian", 'his thoughts' is the theme, 'liberian' is the source, and 'Libertarian' is the goal of the predicate.

2.1.10 Path

Path identifies the argument that denotes the trajectory of the displaced entity, the theme or the agent, in a motion event. For example, 'along the river' is the path in sentence "I walked along the river". The definition of a path depends on the nature of the ground, such as the ground's liquidity ("The knife went inside the pool of chocolate" is meaningless), its countability ("The ant ran between the hamburger" is meaningless), etc., and the nature of trajectory, such as curvature ("I ran around the running track"), boundedness ("The dog ran across the street"), etc.

2.1.11 Locative & Time

Arguments that denote the fixed spatial organizations of events are the locatives of predicates. They can be the site of a predication or its static position, like 'sky' in "The clouds floated in the sky" and 'store' in "My mother works at a store". Time identifies the argument that denotes the time of occurrence of an event in a predication, like 'yesterday' in "I got the physics final exam yesterday".

2.1.12 Manner

Manner identifies the argument that denotes the way in which an event is carried out. Arguments of manner are used to express intensity like 'heavily' in "I knocked the door heavily", speed like 'quickly' in "I ate the meal very quickly", attitude like 'unwillingly' in "I studied all weekend unwillingly", etc.

2.1.13 Reason

Reason identifies the argument that denotes the prior conditions of a predication, like 'fear' in "I ran from fear". Reasons link other events to a predication by means of the motivation of an agent, so they are connected to the intentions of an agent, like 'need to keep fit' in "Bob jogs because of his need to keep fit". Note that, reasons should precede their predications, so the second clause in "Tom is wearing

a tie since he has a job interview this afternoon" is not the reason of the predicate (in fact there are two distinct predicates in this sentence).

2.1.14 Purpose

Purpose identifies the argument that denotes the result or the consequence of a predicate like 'checkup' in "I went to the doctor for a checkup". Though purposes and reasons seem very much alike, they are sharply different in meaning: purposes denote the contextual end points of predications and reasons are the motivational sources of predications. This distinction can be observed from the sentence "I went to doctor because of my checkup" in which 'checkup' denotes the reason.

2.2 Aspect

Events are temporarily situated relations between the entities and aspect defines the way an event is distributed through the time frame in which it happens. In other words, aspect provides information about the internal contour of an event. How languages encode the internal structure of an event can be shown by the following two sentences:

"John ran"
"John was running"

Although both sentences denote the same event that is situated in the past, the ways they located the event in that past time frame are different. The first sentence expresses the motion event as a complete act, and the second one stretches that act into a continuous interpretation. So, aspect operates on an event structure like a mathematical procedure that adds properties to the basic expression to derive new ones ($run+past+extension \rightarrow run+past+continious$). There are four major classes of aspects [9, 5]: *perfective/imperfective*, *telic/atelic*, *punctual/durative*, and *iterative/semalfactive* which are explained in the following sections.

2.2.1 Perfective/Imperfective

The distinction between these two properties is based on the way an event is viewed from the outside of its temporal frame. Perfective aspect construes an event as a complete unit whether or not that event has itself came to an end.

On the other hand, imperfective aspect is associated with events that are viewed as incomplete, nonunitized. The distinction between these two properties can be shown by the following sentences:

"I have written the letter" (*perfective*)
"I was writing the letter" (*imperfective*)

Although the event in the second sentence can be temporarily related with another event ("The phone rang while I was writing the letter"), the same mechanism cannot be applied to the first sentence since perfective events are not internally structured. So, perfective property causes an event to be understood from a conceptual distance as a single unanalyzed whole. It is used when an event's internal complexity is much less relevant to the interpretation that its unitization. Perfectiveness can also be directly encoded through lexicals like the distinction between eat/eat up, fill/fill up, etc. Imperfectives are also compatible with adverbs of manner because they are internally structured, like in "He wrote the letter slowly".

If an event is not used in perfective, languages can encode just one point in the event's time frame instead of directly encoding it as imperfective. Two of such aspectual properties are *inceptive*, way of denoting the initial point of an event like in "We began to talk together", and *terminative*, way of encoding the end point of an event like in "We stopped talking to each other".

2.2.2 Telic/Atelic

This aspectual property identifies the distinction between the events that denote composite acts constructed by a process with a requisite result and other events. Telic events are resultative, and they have built in goals that must be reached in order to be successfully asserted, and necessarily imply previous events. The distinction between telic and atelic events can be shown by the following sentences:

"Bill reached New York" (*atelic*)
"Bill drove to New York" (*telic*)

The first event, although it has a built-in goal, is atelic since it does not identify a process that results in the requisite goal. So, telic events can be defined as processes that exhaust themselves in their consequences, and even they are interrupted, the processes that precede the results hold. Note that, if the event

in the first sentence is interrupted, then its proposition is nullified, but this is not the case for the second sentence.

There are also other criteria that can be applied to identify whether an event is telic or not. For example, telic events are ambiguous with 'almost' in English since they are formed by a process and a result.

"Bill almost reached New York" (*unambiguous*)
"Bill almost drove to New York" (*ambiguous*)
1. 'nearly started the process of driving'
2. 'nearly came to the result (reached New York)'

Atelic events are also sensitive to durative interpretation since they express only the results of events. So, atelic events cannot be used with 'for' in English, which is used to introduce duration.

"Bill reached New York in two hours"
"Bill drove to New York in/for two hours"

2.2.3 Punctual/Durative

Events that are momentary and have no temporal duration are marked as punctual events. On the contrary, events whose time frames are distributed over time are identified as durative events. The distinction between punctual and durative events can be observed in the following sentences:

"Lisa received a letter" (*punctual*)
"Lisa climbed the tree" (*durative*)

Punctual events are sensitive to time phrases that denote some kind of duration, as in sentences "How long did it take for Lisa to receive a letter" and "Lisa received a letter for a while" which are both nonsense. Durative events are sensitive to adverbs of moment like 'at once' in English, but they do not disallow their usage, only their interpretations are changed. For example, the sentence "Lisa climbed the tree at once" refers to the beginning of the process. Languages provide tools that convert punctual events into duratives, like progressivization in English ("John was receiving packages all afternoon").

Both very short events like "The worm inched along" and single undifferentiated acts like "Fred sat" are not thereby punctuals (both have a time duration). Also, even though momentaneous events appear to be goal directed,

momentaneousness does not directly translate into telicity, like the verb 'reach' in English. Also, there is no relation between punctuality and perfectiveness as there is no relation between duration and imperfectiveness. Punctuality and durativity are inherent features of the meanings of events; perfectivity and imperfectivity are means of viewing events.

2.2.4 Iterative/Semalfactive

Many languages make further aspectual distinction with regard to the quantity of an event. Semalfactive events consist of a single act, and iterative events have multiple subevents, or they are repeated, or they are cycled in a time frame. The following sentences show the distinction between semalfactive and iterative events:

"Bob broke the window"	(*semalfactive*)
"Bob broke all the windows"	(*iterative*)

Since the act of breaking is a punctual event, the second sentence must be interpreted as a repetitive act of breaking (plurality of the patient). So, the second event is iterative. Iterative property also indicates the events that have multiple subevents like in "I shook his hand" and represents events that must be conceptualized in a phase like in "The cursor is blinking on the monitor". Note that, all kinds of serial productive events are marked as iterative like "That factory produced twenty F-16 planes last year".

2.3 Tense

Tense is the way that an event is explicitly indexed for a time frame. It is the grammatical or morphological means that languages use to locate an event in time. Events in linguistic expressions are located on an unbounded, unidimensional extent of time outward from a central zero point, the moment of speech. The time is modeled by languages as an ordered scale of precedences and subsequences relative to a baseline. The time line encoded by languages is inflexible and stable. For example, the utterance "I wrote a letter" always refers to an event that occurred prior to the time of speech. So, languages hand down to its speakers certain temporal constants, like past, future, etc. The time line is also imprecise, that is, kinds of times that constitute linguistic time are not very exact. For example, the hours of a day are not grammaticalized in any

language. Instead, the time line is a simple extent and very gross units of time are sufficient to capture temporal notions. So, tense provides the *deictic* properties like 'location in time' and 'relative order' which require a *reference point* for their determination. In contrast, aspect gives the *nondeictic* contours of an event in its time frame [9, 6].

As mentioned, tense reflects a deictic structure with its two deictic points, the contextually situated reference point and the located point, and the direction and the remoteness of the relation between these two points. Tense locates events in the time with respect to a fixed temporal reference point, and then specifies the relation of the event to that temporal center by some direction and remoteness. For example, in "Bob bought a cake" the reference point is the moment of speech, the located point is the event's occurrence time, and the direction is *past*. Languages also encode the degree of remoteness between the two points (the event's occurrence point and the reference point), which can be observed in the following sentences:

"I would get up at 5:00 A.M."	(*distal*, some time ago)
"I just got up"	(*proximal*)

So, the structure of a language's tense system can be defined with four properties:

- *Tense Locus*: the *reference point*
- *Event Frame*: the *located point*
- *Direction*: *precedes*, *coincides*, or *follows*
- *Remoteness*: *distal*, or *proximal*

There are two choices of tense locus that are encoded by languages: *absolute tenses* and *relative tenses*. Absolute tenses take the present moment of speech as the tense locus and assign distance and direction from the speaker as the deictic center. For example, "John will run to the home" denotes the event of running which follows the speaker's present position in time. Relative tenses take some other event or moment as the tense locus, and its usage can be shown with the following sentence:

"The man sitting in the chair was rich"
1. 'the man who was sitting ...'
2. 'the man who is sitting ...'

Observe that, in the example above, 'being rich' is expressed in an absolute tense, but 'sitting' has no inherent temporal reference (the ambiguity presented).

The present moment of speech does not apply to the event 'sitting', it inherits its tense locus from some other event or some other specified time. Absolute tenses are associated with syntactically and semantically autonomous events, and they are overwhelmingly found in the main clause (independent construction). On the other hand, relative tenses are used with events that are dependent on both the meaning and the form of the other events expressed in an utterance, like in subordinate clauses.

There are also two choices of event frame that are encoded by languages: *simple tenses* and *perfect tenses*. Simple tenses, the fundamental tenses, choose a single point on the time line to bear a relation to the tense locus, like in "Andy jumped" and "Andy is jumping". In contrast, perfect tenses select two distinct points other than the tense locus, like in "Tom had seen the movie". Note that, the event 'see' is not only in the past relative to the moment of speech, but also prior to another past event. This third point, which denotes the other event, is called as *time reference*. Perfect tenses require a complex, dual structured event frame. That is, the event frame is to be judged as prior to or temporarily up to a projected reference point other than the moment of speech. So, in usages of perfect tenses, two event frames are evoked in relation to the tense locus.

According to direction and remoteness, languages use two different systems: *vectorial* systems, undifferentiated extension of time from the tense locus, and *metric* systems, division of time line into definite intervals (like tomorrow, next week, etc.). Since the scope of this work covers only the *vectorial* languages, *metric* systems are not explained. Direction in the vectorial systems is a tripartite domain:

- *Past* (*prior to*)
- *Present* (*coincident with*)
- *Future* (*subsequent to*)

Past denotes an undifferentiated temporal extent moving away from the present moment into the already known or completed, and with enough temporal removal into the unknown and hypothetical. As the temporal distance increases, *past* is generally connected with nonactuality, hypotheticality, counterfactuality, and improbability. *Present* denotes an area of time line simultaneous with the moment of speech. *Present* is neither a specific point nor a vector itself, it is an ideal temporal segment that extends in both directions from the present moment. *Present* is connected with on-line activity, actual events, and likelihood of occurrence. It is also used to encode generic and timeless events as well as habituals. Also, incomplete events and events that have some degree of

extensions (states) are sometimes encoded using *present*. *Future* denotes a vector stretching outward from the present moment in an undifferentiated extent into the unknown and unrealized. Since *future* is connected with unknown, it is generally used to encode inception, prediction, intention, potential, volition, supposition, nonactuality, etc.

2.4 Modality

Speakers often qualify their statements with respect to believability, reliability, and general compability with world or accepted facts. The area of semantics that concerns how such qualifications, made by speakers, are encoded by languages is *modality*. So, modality can be defined as the semantic information that is associated with the speaker's attitude or opinion about what is said [9, 27].

Modality signals the relative actuality, validity, believability, etc. of the content of an expression and affects the overall assertability of an expression. For example, in sentence "Apparently, Maria bought another cat", the word 'apparently' denotes the epistemic (state of knowledge) stance of the speaker about the event expressed in the sentence. The speaker, obviously, is not sure about the occurrence of the event when the sentence is uttered, and 'apparently' sets up a belief context, or a possible world. Note that, modality is not only objective measures of factual status, but also subjective attitudes or orientations toward the content of an expression.

Although languages encode some modality phenomena through *modals*, there is no direct relation between them. Modality is a semantic phenomenon that denotes the content of an expression which reflects the speaker's attitude or state of knowledge about a proposition. Modals are grammatical phenomena that encode a set of semantic and pragmatic properties through word inflections and auxiliary words.

The basic denotation of modality is the opposition of actual and nonactual worlds. So, modality is the way a language encodes the comparison of an expressed world with a reference world. Thus, modality is another semantic phenomenon that shows deictic structure with deictic points as the two worlds that are compared. The basic dichtonomy is a scale, and the factual status of a proposition depends on the extent to which two epistemic deicitic points diverge. This divergence is translated into possibility, evidence, obligation, commitment, etc. The deictic structure of modality can be observed in the following sentences:

"John may go"
"John might go"

The first sentence expresses the possibility of John's going in the future. Although the second sentence expresses the same possibility, it is more epistemically removed from the state of affairs. So, the first expression is closer to the real world compared with the second one in the remoteness scale. It can be observed from previous explanations that, there are different types of modalities and five of them, *epistemic*, *expectative*, *deontic*, *volitive*, and *potential*, are explained in detail.

2.4.1 Epistemic Modality

Epistemic modality can be defined as the structural and semantic resources available to a speaker to express judgment of the factual status of a state of affairs. It concerns the truthness of an expression, but the truthness that is relativized to the speaker. So, the scale of the epistemic modality goes from '*someone does not believe that X*' to '*someone does believe that X*'. For example, in sentence "I was planning to go to the school today", the speaker expresses that the event 'going to school' did not occurred (he does not believe the truthness of proposition $go(speaker, school, today)$). In sentence "I heard that Bob cheated in the exam", although the speaker did not expose to the event of cheating (s/he is not sure), s/he asserted the proposition $cheat(Bob, exam)$ with a high probability of occurrence.

2.4.2 Expectative Modality

Expectative modality can be defined as the structural and semantic resources available to a speaker to encode the likelihood of a state of affairs to occur. So, the scale of the expectative modality goes from '*someone does not plans/intends/expects that X*' to '*someone plans/intends/expects that X*'. Considering the same sentence given in the previous section, "I was planning to go to the school today", the speaker expresses the likelihood of occurrence of the event of his/her going to the school (since s/he was planning to do it). In sentence "Most probably, Bob will not be here before 11 o'clock", the speaker expresses that s/he does not expect Bob's arrival before some time. Note that, the speaker's expectation is not exact, can be nullified.

2.4.3 Deontic Modality

Deontic modality expresses the imposition of a state of affairs on individuals, with modality as deixis, the imposition of an expressed world on a reference world. In other words, deontic modality encodes the restriction of possible future states of affairs to a single choice. So, the scale of deontic modality goes from '*someone believes that the performer of an action must not be X*' to '*someone believes that the performer of an action must be X*'. For example, in sentence "You'd better go to a doctor", the speaker tries to restrict the possible kinds of actions that the hearer can perform to only the event of going to a doctor. In sentence "You should not drink cold water after playing football", the speaker tries to make the hearer to exclude a kind of action, drinking cold water, from the state of affairs that can happen after playing football. Note that both sentences are not at the opposite end points of the scale, none of them implies obligation.

2.4.4 Volitive Modality

Volitive modality expresses the preference of a state of affairs in a possible world to become a state of affairs in the reference (real) world. In other words, volitive modality encodes the will of someone about a state of affairs to become real. So, the scale of volitive modality goes from '*someone does not desire that X*' to '*someone desires that X*'. For example, in sentence "Bob wanted to be a mathematician", the speaker expresses Bob's preference to be a mathematician in past (note that expression also contains an epistemic modality that the speaker does not believe in 'Bob is a mathematician'). In sentence "If the decision was left to me, I would not go to that university", the speaker expresses his/her reluctant in going to a specific university.

2.4.5 Potential Modality

Potential modality expresses someone's potency in making a state of affairs in a possible world real in the reference world. In other words, potential modality encodes the effectiveness, potency of an actor on some on-going process and his/her ability to create new state of affairs in the real world. So, the scale of potential modality goes from '*someone is not effective on/capable of X*' to '*someone is effective on/capable of X*'. For example, in sentence "I can afford $300 per month for a house", the speaker expresses that s/he is capable of paying $300 every month. In sentence "Bob did not understand what was going on", the

speaker states that Bob had no effect on the on-going states of affairs.

2.5 Speech-Act

In a speech situation, an utterance causes some kinds of acts to be performed by both the producer and the consumer. One of these acts can be defined as the conveyance of the speaker's intention to the hearer through that utterance. *Speech-Act* concerns the production of linguistic tokens such as questions, commands, promises, etc., under certain conditions with underlying intentions. In other words, intentions of a speaker are delivered through certain grammatical constructions and speech-act identifies the relationship between the intentions and the grammatical constructs.

For example, the sentence "I promise to bring your notes tomorrow morning" is utterred to define a future act of the speaker (bringing the hearer's notes at a specific time) whose performance is not obvious to both the speaker and the hearer. Note that, expression states that the speaker intends to do that act under the assumption that the hearer prefers the speaker doing that act. Utterance of promise places the speaker under an obligation for doing that act. So, given the conditions listed above with the speaker's intention explained, the speech act *promise* is produced with 'X promise to do ...' in English.

Currently, three types speech-acts are used in this work: *declaratives*, *interrogatives*, and *imperatives*. *Declaratives* are used by speakers to convey some kind of information to the hearer and it is the speech-act type which has no special construction in English, all sentences other than the ones with different speech-act types are declarative sentences. So, sentences "I went to the cinema", "I frequently play tennis", and "I am going to study all day tomorrow" are declaratives. There are two types of *interrogatives*: *yes-no questions* and *wh-questions*. Yes-no questions are produced by speakers to learn the truthness of a proposition for which the sentences "Did you have a breakfast" and "Can you ride a bycle" are examples. Speakers use wh-questions to learn a specific participant of a predication which is not known by the speaker. In English, nearly for every thematic role there is a special word in querying that role, like *who* for *agent*. The sentences "Who broke the window" and "When are you going to take your last final" are examples for wh-questions. *Imperatives* are used by speakers to make the hearer to perform some kind of act. The sentences "Open the window" and "Fill in the blanks" are examples of imperatives.

2.6 Attitude

Speakers often qualify the constituents of an expression with respect to their relevance and importance to the meaning that is to be conveyed. *Attitude* concerns how such qualifiactions made by speakers are encoded in languages. Although both modality and attitude encode some qualifications made by the speakers, they reflect different phenomena of languages. Modality is the semantic information that is associated with speaker's opinion about the overall statement or an event expressed in that statement. Attitude is the pragmatic information which covers the modifications of the consituents of a statement, especially the participants of an event, made to assign importance, evaluation, etc., to them.

For example, the sentence "It was Bob who stole the money" has the same propositional meaning with "Bob stole the money", that is $steal(Bob, money)$. The reason for which the first sentence is uttered in a different form from the second one is the speaker's intention to put an emphasis on the agent. That is, the first sentence is used to express Bob as the important participant of the stealing event. Note that, attitude, like modality, has a scaled structure (eg. important, unimportant, irrelevant). There are different types of attitudes and two of them, *evaluative* and *saliency*, are explained in detail.

2.6.1 Evaluative Attitude

Evaluative attitude expresses the way a speaker encodes his/her own point of view about a constituent in an expression. The scale of evaluative attitude varies with the *goodness* that the speaker attaches to that component. High evaluation is attached to the appreciated components, and low evaluation is attached to the components that are disgusted by the speaker. For example, in sentence "He treated me in a bad manner", the speaker expresses his/her low evaluation about the way someone's, denoted by 'he', treatment of him/her.

2.6.2 Saliency Attitude

Saliency attitude is used to define the importance or relevance of a statement's component. The scale of saliency attitude varies with the importance that the speaker attaches to a text component. High saliency is attached to the entities that the speaker wants to be stressed, and low saliency is attached to the entities that the speaker mentions as background. So in sentence "It was yesterday the window was broken by Bob", 'yesterday' is the constituent that is emphasized

and 'Bob' is the component that is mentioned with low relevance.

2.7 Stylistics

The relationship between the producer of an expression and its consumers, and the social and the cultural environment in which the communication takes place generally affects the way that expression is constructed. Producers take into account their knowledge about the consumers and the social context when they utter expressions and this information is reflected in lexical choices, grammatical structures used, etc. Stylistics is the branch of pragmatics that involves in exploring how conveyance of meaning depends on these two contextual information. For example, consider the following sentences:

> "Could you please open the window"
> "Open the window"

Although both sentences' structures are used to make a consumer to perform a certain act, the way how this meaning is presented to the consumer radically differs. The first sentence is generally uttered in a formal situation, and in the second one the situation is such that the producer is in a higher statue compared with the consumer. Note that, stylistics reflects the structure of the relationship between humans, so it is also defined on a scale. This structure can be demonstrated by the sentence "Can you open the window" which defines a situation between the two extremes given as examples above. Stylistics can be analyzed in six different subtopics: *formality*, *respect*, *politeness*, *simplicity*, *color*, and *force*.

Formality scales situations from cases in which there is no specific relationship between the producer and the consumer, like a dialogue between the representatives of two countries, to cases in which the producer and consumer knows each other very well and have a sincere relationship, like the conversation between very close friends.

Respect scales situations from cases in which the relationship between the producer and the consumer is well defined according to social and cultural status of them and the opinions of one is very important for the other to cases in which both the producer and the consumer do not take care the other.

Politeness scales situations from cases in which behaviors and requests of the producer and responds of the consumer are well defined and restricted by social

and cultural context to cases in which the context is irrelevant (no restrictions) to the dialogue that is made between the producer and the consumer.

Simplicity scales situations from cases in which the exchange between producer and consumer is not restricted by any information context, like conversation between two expert doctors about the diagnosis of a patient, to cases in which producer tries to explain a phenomenon that is outside the knowledge of the consumer, like the conversation between a doctor and his/her patient.

Color scales situations from cases in which the producer tries to decorate the things s/he wants to be conveyed in an impressive way through defining an imaginary world, exaggarated feelings, etc., like in poems and novels, to cases in which information exchange between the producer and the consumer is the only purpose, like in technical reports.

Force scales situations from cases in which the producer has the power to make the consumer to perform a certain act, like the prohibition of smoking of a doctor to his/her patient, to cases in which the producer has no control on the behaviors and thoughts of the consumer.

Chapter 3

Knowledge Resources & Representation Languages

The goal to be achieved in this thesis, as mentioned, is to design a prototype system that generates the feature structure representation of a natural language sentence from its language independent representation. The language independent representation is called as *text meaning representation* (TMR) which is developed for the Microcosmos project at New Mexico State University [18, 3, 19]. To achieve such an independent representation, two resources of knowledge are utilized: *speaker's world knowledge* about entities, events, their relationships and interactions, and *linguistic information* about semantic (aspect, thematic roles, modality, etc.) and pragmatic (speech-act, stylistic factors, etc.) issues explained in Chapter 2. Also, additional information about the overall situation (relations between events, references to entities, time references, etc.) is provided in TMR representation whenever it is appropriate. The feature structure (f-structure) representation of a sentence is used to encode the syntactic properties of that sentence such as open-class and closed-class lexical items to be used, verbal phrases, grammatical roles, noun phrases, and other complex structures.

The generation system requires introduction of lexical items and mapping between the structures of TMR and f-structure representations. To handle such a task, the designed system uses four knowledge resources: *ontology*, *lexicon*, *map-rules*, and em f-structure representation of the target language. *Ontology* is a hierarchical representation of speaker's world knowledge about entities, events, and their relationships in an abstract way. The knowledge that is provided in ontology is language independent. *Lexicon* contains information about the relationship between open-class lexemes of the target language and abstract entries (concepts) of the ontology which are used in TMR. This relationship

is achieved through defining a lexeme in the lexicon by limiting the abstraction provided by an ontology entry. Lexicon also contains information about semantic and pragmatic properties of open-class lexemes and usages of closed-class words. *Map-rules* define how the content of a TMR is related to the syntactic structure of the target language. They encode how available information is extracted from a TMR and how such information updates the syntactic structure of a sentence. *F-structure representation* of the target language is given as a seperate knowledge resource to avoid any language-dependent information inside the system.

To reach a complete understanding of how these representation languages (TMR and f-structure) and three knowledge resources are related with each other, each of them is analyzed individually, starting from more abstract notions to language specifics. First, ontology and text meaning representation are described, then utilized f-structure representation of the target language, Turkish in our case, is presented, and finally two knowledge resources, lexicon and map-rules, that provide the interface between TMR and the target language are explained in detail.

3.1 Ontology

Natural language expressions are produced to convey some information, held by producers, about entities and events of the world, including relationships hold among them and interactions occur between them. So, to represent the meaning of an expression in a language neutral way, an abstract model of the world is needed. *Ontology* is the computational model that is designed for meeting this need [2, 17, 16]. It is the knowledge resource that provides general information about the world in a hierarchical way like a human-being realizes the world. Note that, ontology does not contain any information which is specific to any human-being.

Every entry, called a *concept*, in the ontology is a primitive symbol that represents a proposed abstraction about a set of things in the world. It captures their common properties and their relations with other concepts. Each concept is represented by a frame and knowledge is encoded through *feature-value* pairs and *slots*. *Feature-value* pairs are used to encode the properties of a concept. *Slots* are special constructions and they are utilized to group feature-value pairs that describe the aspects of a concept's general property.

Each concept represents either a group of entities or a set of similar events.

A concept which is created for a group of entities decomposes the definition into a set of properties that any entity from that group can take. Each property takes its values from a well defined domain and representation of real world entities is achieved through instantiating these properties with specific values. For example, humans can be represented with the following simplified frame:

$$\begin{bmatrix} concept & HUMAN & \\ definition & \begin{bmatrix} type & common/proper \\ name & human\text{-}names \\ gender & male/female \\ age & \geq 1 \ \& \leq 120 \\ job & teacher/engineer/\ldots \end{bmatrix} \end{bmatrix}$$

In this example, humans are defined by only five properties: *type*, *name*, *gender*, *age*, and *job*. *Type* property is introduced to make a distinction between humans whose names are known by the speaker and others. So, if the property *type* takes the value *common*, then the property *name* is undefined. To show how a real human, Ali, a male at the age of 25 who is a computer engineer, is represented by such an abstraction, the following instantiated *HUMAN* frame is given.

$$\begin{bmatrix} concept & HUMAN & \\ definition & \begin{bmatrix} type & proper \\ name & Ali \\ gender & male \\ age & 25 \\ job & computer\text{-}engineer \end{bmatrix} \end{bmatrix}$$

Representation of events with concepts is somehow different from representation of entities in the ontology, since they are like predications over arguments. So, a concept which is created for a set of similar events contains the argument structure of those events under *roles* slot, besides definitions of their properties. *Roles* slot defines all possible thematic roles that set of events can take. For example, events that describe some motion of an actor from one location to location can be represented by the following frame:

$$\begin{bmatrix} concept & ACTOR\text{-}MOTION & \\ roles & \begin{bmatrix} agent & ANIMATE \\ source & LOCATION \\ destination & LOCATION \\ instrument & ARTIFACT \end{bmatrix} \end{bmatrix}$$

Four arguments are defined for an actor-oriented movement: *agent*, *source*, *goal*, and *instrument*. Observe that, the values of an event's arguments can be limited to other concepts from the ontology, like value domains used in the definition of an entity. Following example is given to show how a real event, the movement of a human, Ali, from Ankara to İstanbul in an airplane, is represented by instantiating *ACTOR-MOTION* concept defined previously.

$$\begin{bmatrix} concept & \text{ACTOR-MOTION} & \\ roles & \begin{bmatrix} agent & HUMAN(Ali) \\ source & CITY(Ankara) \\ destination & CITY(\dot{I}stanbul) \\ instrument & VEHICLE(airplane) \end{bmatrix} \end{bmatrix}$$

Although the concepts in the examples above are named with English words, they are not simple mappings of those words' senses into renamed entries in the ontology. The ontology is a language independent world modeling such that a concept can represent a set of lexemes of any language. For example, *HUMAN* can be used to denote the nouns *John*, *man*, *woman*, *girl*, etc. in English, and *ACTOR-MOTION* can be used to represent the verbs *go*, *come*, *reach*, etc. in English.

The whole ontology is constructed through connecting individual concepts by a set of relations. The main relation, *is-a*, forms an inheritance mechanism in the ontology. It constructs a concept hierarchy that is determined according to the abstraction a concept provides. That is, a concept that defines a subset of entities or events covered by another concept is connected to that concept with an *is-a* link. The child concept provides additional information that constrains the abstraction defined in the parent concept. In this way, enumeration of knowledge in one level representation is avoided, the common properties are encapsulated by parent concepts. Also with this relation, decomposition of interpretation is achieved, which is similar to the way humans realize the world. Note that, a concept can inherit from more than a parent concept, which forms a multi-parent tree structure in the ontology.

The general structure of an imaginary ontology is shown in Figure 3.1. Note that the root, *ALL*, has two child concepts, *ENTITY* and *EVENT*, that define the two main categories used in the representation of the world knowledge. The inheritance mechanism utilized in the design of the ontology is shown for both entity and event concepts. The entity *ANIMATE* covers all animals in the world with common properties *gender*, *height*, *weight*, etc. *HUMAN*, a kind of *ANIMATE*, has additional properties like *type*, *name*, *job*, etc. The event

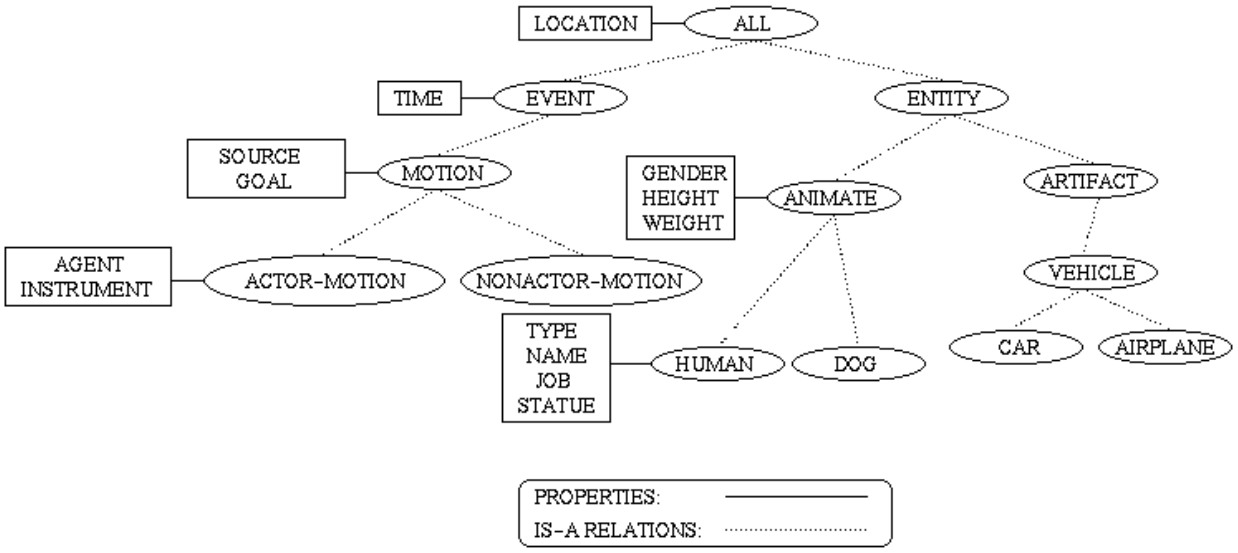

Figure 3.1: An Imaginary Ontology Structure

MOTION represents all kinds of events that somehow contain a movement and it has common thematic roles *source* and *goal*. *ACTOR-MOTION* identifies *MOTION* events that are performed deliberately by some actor and have additional arguments *agent* and *instrument*. Also observe that, even the most abstract concepts, like *EVENT* with *time* and *ALL* with *location*, provide the common properties of its child concepts.

The concepts in the ontology are also related with each other through a variety of other relations. These links do not impose an inheritance mechanism, but allow to define specific relationships that exist between concepts. For example, *is-part-of* is used to encode the relationship between the constituents of an entity and the entity itself. For example, a monitor is a constituent of a computer, so its definition should be as follows:

$$\begin{bmatrix} concept & MONITOR \\ definition & \begin{bmatrix} : & : \\ : & : \end{bmatrix} \\ is\text{-}part\text{-}of & COMPUTER \end{bmatrix}$$

Such links are defined whenever appropriate to make general inferences about the relations between concepts or to fill gaps in expressions that are supposed to be completed by the text consumers.

A developed ontology can be utilized in several ways in an interlingua MT system and three of them are used in this work. Its first usage in this work, as mentioned, is in text meaning representation. Besides using linguistic

information, the TMR of a natural language expression is constructed through defining the entities and the events that are referenced in that expression, and these entities and events are represented by concepts from the ontology. Since a concept is a generic representation of a set that have similar properties, the representation of real world entities and events in a TMR is achieved through instantiating the features used in the definitions of those concepts. Two previously given examples, $HUMAN$ and $ACTOR\text{-}MOTION$, explain how concepts are utilized in text meaning representation.

Ontology is also utilized in the design of the generation lexicon. Since the words of a language generally encode entities and events of the world, the definition of a word sense is made by using of a concept. Since concepts in the ontology are generic entities or events, the definition of a word sense should limit the abstraction provided by a concept through constraining the value domains of the features and excluding some of the properties of that concept (remember that concepts are not mappings of word senses). So, the definition of a word sense is made by instantiating a concept in the ontology. Observe that, instantiation is used in both text meaning representation and generation lexicon, and this builds the connection between the open-class lexical items of the target language and the representation provided in a TMR.

Finally, ontology is used in the design of the map-rules. Since an instantiated concept in a TMR inherits the definitions of all its ancestor concepts. That is, the features of the ancestors can be used in the instantiation, the mapping between TMR and f-structure representation should be done in a way such that it follows this hierarchical structure. So, mappings that are common to all children of a concept are associated with that parent concept. In other words, the applicable map-rules of a concept are collected from concepts which are on the path from that concept to the root in the ontology tree.

There are also some general advantages in developing an ontology for machine translation systems and the important ones are given in the following list [16]:

- Ontology enables an MT system to share knowledge between analysis and generation lexicons, since it is an interface between the two processes. It also eliminates the need for bilingual dictionaries between language pairs in a multi-lingual translation environment. Analysis and generation lexicons are constructed only once for an MT system that utilizes an ontology in this way.

- Ontology can be utilized to fill meaning gaps in expressions. Situations in which producers do not provide some specific information explicitly, but instead suppose their consumers to infer that information, are very common in natural language use. In such cases, an MT system can make inferences by using the relations defined between concepts (*made-of*, *is-part-of*, etc.) in the ontology which improves the quality of text understanding (in analysis phase).

- Ontology can also be used to resolve semantic ambiguities that reside in expressions by making inferences through utilizing the topological structure (*is-a*) and the content of the ontology itself. For example, whether an event has semalfactive or iterative interpretation can be resolved by using that event's time properties (whether it is punctual or not, etc.). Remember that, a punctual event in progressive tense requires an iterative interpretation.

3.2 Text Meaning Representation

In interlingua machine translation approach, the translation between the source and the target language is achieved through describing the meaning conveyed in sentences of the source language in an intermediate representation which is language-independent [24, 15, 24, 23]. The intermediate representation which is used in this work is taken from the Microcosmos project [19, 18, 3] and it is called text meaning representation, or shortly TMR. To get such a general representation, various knowledge resources are utilized: ontology (language-independent world knowledge), semantic properties (temporal relations, aspectual properties, modality information, etc.), and pragmatic information (speech-acts, speaker's attitude, and stylistics factors). Also, there are a few special constructs which are used to handle some phenomena in language that are not covered by the above resources, such as time references, entity references, and sets.

To achieve language independence in TMR, no specific information about the source language is included inside TMR such as that language's lexical items and syntactic structures. So, syntactic information such as a sentence's verbal and noun phrases, its tense, its grammatical roles, and its word ordering are avoided in this representation. Instead, the resources mentioned above are used to capture the meanings of individual elements and their relationships. In this way, both the propositional and non-propositional meaning expressed in a source sentence can be represented.

Text meaning representation is a formal, frame-based language in which the meanings of open-class lexical items are represented by instantiating concepts from the ontology. But, instantiations of concepts are not enough to capture the overall meaning resides in a sentence. So, information about the semantic and the pragmatic properties of a sentence and the relations between the components should also be described. To facilitate this, the TMR language contains special notations for representing aspectual properties, speaker's attitudes, modalities, speech-acts, etc. The followings are the list of frames that are used in meaning representation, and each frame is explained and exemplified in their usage in the rest of this section.

- Table-Of-Contents
- Instantiated Concepts
- Time Frame
- Temporal Relations
- Aspect Frame
- Modality Frame
- Attitude Frame
- Speech-Act Frame
- Coreference Frame
- Focus Frame
- Set Frame
- Domain Relations
- Stylistic Information

3.2.1 Table-of-Contents

This frame type is used for providing a summary of propositions, relations, and discourse information about the overall sentence. It is used to extract general knowledge about the representation in hand without searching for frames in TMR. It is filled after all the frames in a TMR are created, and it contains knowledge about the following frames in a TMR: the list of events, temporal relations, attitudes, modalities, focuses, coreferences, domain relations, the speech-act and the stylistic information frames. Its corresponding structure is the following:

$table\text{-}of\text{-}contents_i$

$speech\text{-}act$	$speech\text{-}act_i$
$heads$	$list\text{-}of(EVENT_i)$
$temporal\text{-}relations$	$list\text{-}of(temporal\text{-}rel_i)$
$attitudes$	$list\text{-}of(attitude_i)$
$modalities$	$list\text{-}of(modality_i)$
$focuses$	$list\text{-}of(focus_i)$
$stylistic\text{-}factors$	$stylistics_i$
$coreferences$	$list\text{-}of(coreference_i)$
$domain\text{-}relations$	$list\text{-}of(domain\text{-}rel_i)$

3.2.2 Instantiated Concepts

This type of frames are constructed through setting some of the properties of concepts defined in the ontology to make the connection between the real world and the meaning representation. As explained, there is a major classification in the ontology that splits the concepts into two main categories: events and entities. This distinction extends to the representation in TMR. Events are used to denote propositions expressed in a sentence, so they require extra information about their aspectual properties, their time of occurrences, and theie propositional truthness assigned by the producer, in addition to the definition provided by the ontology. This extra information is represented through *aspect* and *time* slots, and *polarity* feature is used to denote the expressed judgment about the truthness of the expression. This feature takes either *positive* or *negative* value. Entities correspond to the arguments of the expressed propositions, so instantiations are made by just filling a set of features which are provided in the definitions of those concepts.

$READ_i$		$HUMAN_i$	
agent	$HUMAN_i$	*type*	*common*
theme	$BOOK_i$	*gender*	*female*
aspect	$aspect_i$	*age*	≥ 17
time	$time_i$		
polarity	*positive*	$BOOK_i$	
		type	*fiction*

The incomplete TMR given above is used to represent the proposition $read(woman, fiction\text{-}book)$. Observe that, the connection between the proposition and its arguments is made through thematic roles *agent* and *theme*, and the required additional information about aspect, time, and polarity are given in the event $READ_i$. A woman is represented by a $HUMAN_i$ whose gender is female and whose age is greater than 16, and a fiction book is defined by a $BOOK_i$ with its type as fiction.

3.2.3 Time Frames

This type of frames are used for two reasons. First, they are utilized for relating the events in an expression temporally, including the moment of speech. In this usage, the contents of time frames are irrelevant to the meaning representation since such frames are used to relate the occurrences of events in the time line through temporal relations (tense in Chapter 2). So, only a dummy information

is provided with *absolute* feature, which is utilized in processing time frames. The following is the structure of time frames when they are used for this reason.

$time_i$
  absolute *past/present/future*

Second, they are used to provide exact time references that are made in an expression. In other words, this usage is utilized whenever the time of occurrence of an event is mentioned explicitly in an expression. There are three types of time references that are handled in this work:

• *Fixed Times* : "I will go to Istanbul on *Sunday.*", "I have an exam at *13:40.*"

• *Durations* : "I have been working *for two hours.*"

• *Intervals* : "The school was built *between 1985 and 1988.*"

The following frame structures are the proposed representations of these time references:

$time_i$	
day	*sunday/.../saturday*
date	$1/\ldots/31$
month	*january/.../december*
year	$0/\ldots/2000$
hour	$1, 1:30, 1(am), 1(pm)$

$time_i$	
duration	$day, week, \ldots, hour$
unit	*integer*

$time_i$	
beginning	$day, \ldots, hour$
end	$day, \ldots, hour$

3.2.4 Temporal Relations

This type of frames are used to represent the relations between the time of occurrences of events that are expressed in a sentence. They are utilized to find the tense (in Chapter 2) of a sentence which is to be generated. This type of frames, like other relation frames, has two arguments which are time frames, and a relation type that can take one of the following values:

• *After*: Relates two events that do not interleave in the time line (for *past* and *future*). For example, this relation type is used for the sentence "I went to the cinema" which is uttered after the occurrence of the real event.

• *At*: Relates two events that occur at the same time. This relation type is used for sentences such as "While I was studying, he was listening music very loudly".

• *During*: Relates two events such that one event occurs in a time interval which is captured by the duration of the other event. In sentences such as "When you phoned me, I was watching TV.", this relation type is used.

• *Extend*: This relation type is utilized whenever the relationship between the two events are indeterminate, like the present tense usage in English. (in "I frequently go to the city library", the event can take place before, after, and even at the moment of speech)

The following is the structure of the temporal relation frames:

$temp\text{-}rel_i$

type	$after/at/during/extend$
arg_1	$time_j$
arg_2	$time_k$

3.2.5 Aspect Frames

This type of frames are used to define the aspectual properties (defined in Chapter 2) of every event, except the speech-act, in an expression. They provide knowledge that can be utilized in lexical selection, syntactic marking and tense determination. The following is the structure of aspect frames:

$aspect_i$

phase	$perfect/begin/end/continue$
duration	$momentary/prolonged$
iteration	$single/multiple$
telicity	$true/false$

In cases when the value of a feature in an aspect frame cannot be determined, *unknown* filler is used. Two examples are given to show how the aspectual properties of an event are represented in a TMR.

"He broke the windows"

$aspect_i$

phase	$perfect$
duration	$prolonged$
iteration	$multiple$
telicity	$true$

"I am going to the school"

$aspect_j$

phase	*begin*
duration	$prolonged$
iteration	*single*
telicity	$false$

3.2.6 Modality Frames

This type of frames are used to represent information about someone's opinion on an event or an entity that is expressed in a sentence. A modality frame is defined through four features: *type*, *value*, *scope*, and *attribution*. *Type* takes its value from the modality types described in Chapter 2 and *value* is a kind of scaling about the strength of that opinion. *Scope* refers to the entity or the event on which the opinion is held and *attribution* refers to the human who has that opinion. A modality frame can also take a time frame in its definition which provides an extra information about the time at which the opinion is held. The structure of modality frames is shown below:

$modality_i$	
type	*epistemic/deontic/volitive/potential/expectative*
value	[0..1] (*real*)
scope	*CONCEPT*
attribution	*HUMAN/speaker/hearer*
time	$time_j$

The following two examples show how the modality information defined in Chapter 2 is used in text meaning representation.

"You should go to a doctor"

$modality_i$	
type	*deontic*
value	≥ 0.80
scope	GO_i
attribution	*speaker*

"He is supposed to be here"

$modality_j$	
type	*expectative*
value	1
scope	$HUMAN_i$
attribution	*speaker*

3.2.7 Attitude Frames

This type of frames are used to encode someone's attitude toward an entity or an event expressed in a sentence. The structure defined for modality frames is also used in the representation of attitude frames. Its *type* takes one of the two values, *saliency* or *evaluative*, that are also defined in Chapter 2. Two examples given below show the usage of attitude frames in TMRs.

"It was that boy who broke the window."

$attitude_i$	
type	*saliency*
value	1
scope	$HUMAN_i$
attribution	*speaker*

"The circumstances are worse than ever."

$attitude_j$	
type	*evaluative*
value	0
scope	$CIRCUMSTANCE_i$
attribution	*speaker*

3.2.8 Speech-Act Frames

The information about the speech situation is given by using this type of frames. Currently, a speech situation is defined by five features: *type* (speech-act type: *declarative*, *interrogative*, or *imperative*), *scope* (the reason for which the expression is produced), *producer* and *consumer* (contributors of the situation), and *time* (its time of occurrence). The structure of speech-act frames is the following:

$speech\text{-}act_i$	
type	$declarative/interrogative/imperative$
scope	$CONCEPT$
producer	$HUMAN/speaker/author$
consumer	$HUMAN/hearer/reader$
time	$time_i$

The distinction between written and spoken expressions is handled by *producer* and *consumer* features that can provide extra information about the stylistic issues. The time of the speech is utilized in determining the temporal relations between the events expressed and the moment of speech. Following two examples are given for explaining the usage of speech-act frames:

"Stop watching TV."	
$speech\text{-}act_i$	
type	*imperative*
scope	$STOP_i$
producer	*speaker*
consumer	*hearer*
time	$time_i$

"I bought a new cassette."	
$speech\text{-}act_j$	
type	*declarative*
scope	BUY_i
producer	*speaker*
consumer	*hearer*
time	$time_j$

Speech-act frames can also take modality and attitude frames to represent opinions and attitudes held in the time of the speech about the entities and the events referenced in the expression.

3.2.9 Coreference Frames

Referring to entities without explicit definitions is a common phenomenon in natural languages. Coreference frames are utilized to handle references in texts. They are also used to avoid enumeration of instantiated concepts and time frames in meaning representation. The following is an example about the usage of coreference frames:

"Mary asked for that book and took it with her."

ASK_i		$TAKE_i$	
agent	$HUMAN_i$	*agent*	$HUMAN_j$
theme	$BOOK_i$	*theme*	$BOOK_j$
⋮	⋮	⋮	⋮

$coreference_i$ $HUMAN_i$, $HUMAN_j$
$coreference_j$ $BOOK_i$, $BOOK_j$

3.2.10 Focus Frames

This type of frames are used in TMR to represent expressions of emphasis. For example, in a passive construction, although the propositional content does not change, the emphasis of the expression is changed. It can also be used for handling free word ordering phenomenon in languages such as Turkish. It has two features, the first one denotes the frame on which the emphasis is put, and the second one represents the degree of emphasis. Its usage is exemplified by the following incomplete TMR:

"He was punished by the manager for being late."

$PUNISH_i$		$speech\text{-}act_i$	
agent	$HUMAN_i$	*type*	*declarative*
patient	$HUMAN_j$	*scope*	$PUNISH_i$
⋮	⋮	⋮	⋮
$focus_i$		*focus*	$focus_i$
scope	$HUMAN_j$		
value	1		

In the example above, $HUMAN_i$ represents the manager and $HUMAN_j$ denotes the person who was punished. Without the $focus_i$ frame, the sentence is realized as "The manager punished him for being late.". So, representation of the emphasis on the *patient* in this sentence is achieved using $focus_i$.

3.2.11 Set Frames

This type of frames are used to represent a broad range of phenomenon such as definite and indefinite sets, ordinals, superlatives, and existentials. A set frame is defined with four features: *member-type*, *cardinality*, *members*, and *excludes*. The *member-type* feature can be a concept like $STUDENT$, or an instantiated concept to constrain the set into a more specific one such as $STUDENT$s who

take physics courses. Its *cardinality* denotes the number of entities which belong to that set, and its value can be either exact (2) or an interval (≥ 4). Some of the members can be enumerated in *members* feature for representing sets like $STUDENT$s who take physics courses including John, Marry, and Charles. Also, some of the entities who satisfy the set properties can be excluded using *excludes* feature for denoting sets like $STUDENT$s without Erdem, Ayşin, and Evrim. So, the following is the structure of set frames used in this work:

set_i

member-type	$CONCEPT$
cardinality	*integer/range*
members	$CONCEPT$
excludes	$CONCEPT$

The following examples show how set frames are utilized in TMR to handle some of the phenomena mentioned previously.

"They went to the cinema."

GO_i

agent	set_i
destination	$LOCATION_i$
⋮	⋮

set_i

member-type	$HUMAN$
cardinality	> 1
excludes	*speaker, hearer*

"There are two apples on the table."

$FRUIT_i$

type	*apple*

set_j

member-type	$FRUIT_i$
cardinality	2

$speech\text{-}act_i$

type	*declarative*
scope	set_j
⋮	⋮

"Take all the books other than these two."

$TAKE_i$

agent	set_i
theme	set_j
⋮	⋮

set_i

member-type	$HUMAN$
cardinality	≥ 1
members	*hearer*
excludes	*speaker*

set_j

member-type	$BOOK_i$
cardinality	> 2
excludes	set_k

set_k

member-type	$BOOK_i$
cardinality	2

3.2.12 Domain Relations

This type of frames are used to represent explicit connections between events or entities. They are introduced to provide:

- Dependence between constituents (since, although, etc.).
- Adjacency between constituents like conjunctions (and, or, etc.).
- Relations between constituents like exemplification (such as, like, etc.).

Like temporal relations, domain relation frames take two arguments and relate them by a domain relation type. The structure of domain relation frames is as follows:

$domain\text{-}relations_i$	
$type$	$reason/enumeration/particular/exclusive\text{-}or/etc.$
arg_1	$CONCEPT$
arg_2	$CONCEPT$

The examples below explain the usage of domain relations in TMR.

"Since Ali didn't study enough, he didn't pass the exam."

$STUDY_i$		$coreference_i$	
$agent$	$HUMAN_i$	$HUMAN_i, HUMAN_j$	
⋮	⋮	$domain\text{-}relation_i$	
$PASS_i$		$type$	$reason$
$agent$	$HUMAN_j$	arg_1	$STUDY_i$
$theme$	$EXAM_i$	arg_2	$PASS_i$
⋮	⋮		

"I will either go to the cinema, or stay at home."

GO_i		$STAY_i$	
$agent$	$HUMAN_i$	$agent$	$HUMAN_j$
$destination$	$LOCATION_i$	$location$	$HOME_i$
⋮	⋮	⋮	⋮
$coreference_j$		$domain\text{-}relation_r$	
$speaker, HUMAN_i, HUMAN_j$		$type$	$exclusive\text{-}or$
		arg_1	GO_i
		arg_2	$STAY_i$

3.2.13 Stylistics Frame

Situations in which expressions are produced generally affect lexical selection and grammatical construction. For example, usage of slang words is inappropriate

in technical writing. All such information is provided by stylistics (described in Chapter 2) frame in TMR which defines various aspects of the stylistic information that can be used in generation phase. The following is the structure of stylistics frames:

$stylistic_i$	
$formality$	[0..1] ($range$)
$respect$	[0..1] ($range$)
$politeness$	[0..1] ($range$)
$simplicity$	[0..1] ($range$)
$color$	[0..1] ($range$)
$force$	[0..1] ($range$)

3.2.14 A TMR Example

To show how a TMR is constructed for a given expression, the following Turkish sentence is analyzed and its corresponding TMR is constructed with detailed explanation:

"Kitap okuyan kadına bir elma verecektik"
("We were going to give an apple to the woman who was reading a book")

There are two events in the sentence above: $GIVE$, the main event, and $READ$, which gives additional information about the woman. The construction starts with the main event, $GIVE$, which has three arguments. Its *source* is a set of $HUMAN$ that includes the *speaker*, its *destination* is also a $HUMAN$ whose *gender* is $female$ and *age* is greater than 16, and its *theme* is a $FRUIT$ (note that reference to *theme* is unimportant). The *speaker* refers to the beginning of the event (*phase*,*begin*), and the event is punctual and there is no repetition [(*duration*,*momentary*), (*iteration*,*single*), (*telicity*,$false$)]. The *source* has an expectation that, the event will occur with a high probability ($modality_1$).

$GIVE_1$	
$source$	set_1
$destination$	$HUMAN_1$
$theme$	$FRUIT_1$
$polarity$	$positive$
$aspect$	$aspect_1$
$time$	$time_1$
$modality$	$modality_1$

$FRUIT_1$	
$type$	$apple$
$reference$	$indefinite$
$aspect_1$	
$phase$	$begin$
$duration$	$momentary$
$iteration$	$single$
$telicity$	$false$

set_1	
member-type	$HUMAN$
cardinality	≥ 2
includes	*speaker*

$HUMAN_1$	
type	*common*
gender	*female*
age	≥ 17
reference	*definite*

$time_1$	
absolute	*past*

$modality_1$	
type	*expectative*
value	≥ 0.75
scope	$GIVE_1$
attribution	set_2
time	$time_2$

Then, the construction continues with the event $READ$, which has only two arguments. Its *agent* is the woman who is the *destination* of $GIVE$, and its *theme* is a $BOOK$ (note also that reference to the *theme* is unimportant). The *speaker* refers to a midpoint of the event's time frame (*phase*,*continue*), and the event is a process and there is no repetition [(*duration*,*prolonged*), (*iteration*,*single*), (*telicity*,*true*)].

$READ_1$	
agent	$HUMAN_2$
theme	$BOOK_1$
polarity	*positive*
aspect	$aspect_2$
time	$time_3$

$BOOK_1$	
reference	*indefinite*

$aspect_2$	
phase	*continue*
duration	*prolonged*
iteration	*single*
telicity	*true*

$time_3$	
absolute	*past*

Next, the information about the speech situation is encoded. The whole sentence is an assertion (*type*,*declarative*), and its scope is the event $GIVE$. At the time of the speech, the *speaker* knows that the event $GIVE$ did not occur ($modality_2$).

speech-act$_1$	
type	*declarative*
scope	$GIVE_1$
speaker	*speaker*
hearer	*hearer*
time	$time_4$
modality	$modality_2$

$modality_2$	
type	*epistemic*
value	0
scope	$GIVE_1$
attribution	$HUMAN_3$
time	$time_5$

$time_4$	
absolute	*past*

The temporal relations between the expressed events and the moment of speech must be defined next. The event $GIVE$ is about to occur during the

event $READ$ ($temp\text{-}rel_1$), and the sentence is produced after the event $READ$ ($temp\text{-}rel_2$).

$temp\text{-}rel_1$		$temp\text{-}rel_2$	
$type$	$during$	$type$	$after$
arg_1	$time_1$	arg_1	$time_4$
arg_2	$time_3$	arg_2	$time_1$

Note that, there are frames that are not filled in previous parts due to their equivalence with other defined frames. This information is given through $coreference$ frames. The expectation ($modality_1$) is held by the *source* of the event $GIVE$ ($coreference_1$) and it is at the same time with that event ($coreference_2$). As mentioned, the *destination* of $GIVE$ is the *agent* of $READ$ ($coreference_3$). The belief ($modality_2$) is held by the *speaker* ($coreference_5$) and it is at the same time with speech ($coreference_4$).

$coreference_1$	set_1, set_2
$coreference_2$	$time_2$, $time_1$
$coreference_3$	$HUMAN_1$, $HUMAN_2$
$coreference_4$	$time_5$, $time_4$
$coreference_5$	$speaker$, $HUMAN_5$

After these phases, the last frame, *table-of-contents*, is filled with a summary about the major frames in this TMR and the construction process is finished.

tc_1	
$speech\text{-}act_1$	$speech\text{-}act_1$
$heads$	$GIVE_1$, $READ_1$
$temp\text{-}rels$	$temp\text{-}rel_1$, $temp\text{-}rel_2$
$modalities$	$modality_1$, $modality_2$
$attitudes$	NIL
$sets$	set_1, set_2
$focuses$	NIL
$stylistics$	NIL
$coreferences$	$coreference_1$, $coreference_2$, $coreference_3$, $coreference_4$, $coreference_5$
$domain\text{-}rels$	NIL

3.3 Feature Structure Representation

Although meaning in natural language expressions is at the core of the knowledge-based, interlingua MT methodology that is utilized in this thesis, the way a

specific language encodes meaning syntactically is also central to a generation system. Syntactic encoding of a language covers how the distinction between the arguments of a verb is made by using case markings and word order, how auxiliary verbs and inflectional endings are used to represent tense and aspect of a sentence, and how noun phrases are constructed.

Since the ultimate goal is to produce target sentence from TMRs, the interface between meaning representation and target language's syntactic structure should be achieved, and a sentence's syntactic structure is represented by using a special formalism, called *feature structure*, in this work. The system developed and implemented in this work, as mentioned, produces the feature structure representations of target language sentences from input TMRs. The f-structure formalism of Turkish, which is our test language, is taken from Hakkani's thesis [11]. Our system produces an output that can be fed into Hakkani's tactical generator to generate the final Turkish sentence.

Feature structure representation is used to cover the syntactic properties of a sentence for a specific language. This representation also contains the lexical items to be used in the final sentence. F-structure representation is also a frame-based formalism and have two kinds of constructions. The first one, $feature$, is the minimal unit of this representation and it is only formed by feature name and a value from a predefined domain,. Features are used to represent the names of syntactic properties such as tense, voice, etc. The second one, *slot*, is used to represent grammatical functions such as a sentence's verb, its subject, etc. A slot contains a set of $feature$-$value$ pairs and other *slots* that are constituents of that function. So, syntactic properties that are used to describe a general syntactic construction are grouped under a *slot*. The general structure of f-structure representation is shown in Figure 3.2:

$$\begin{bmatrix} feature_1 & value_{1.1}/\ldots/value_{1.n} \\ \vdots & \vdots \\ feature_i & value_{i.1}/\ldots/value_{i.m} \\ \\ slot_1 & \begin{bmatrix} \vdots & \vdots \end{bmatrix} \\ \vdots & \vdots \\ slot_j & \begin{bmatrix} \vdots & \vdots \end{bmatrix} \end{bmatrix}$$

Figure 3.2: Frame-Based Representation of F-Structure

Syntactic structure of Turkish sentences can be analyzed in three main

constructional categories [11, 28]: *simple sentences*, *complex sentences*, and *noun phrases*. The f-structure representation of Turkish sentences used in this work uses these major categories. The first category, *simple sentences*, covers expressions like "Kadın camı kırdı" ("The woman broke the window"), "Ali kitap okumak istedi" ("Ali wanted to read a book"). Complex sentences are differentiated from simple sentences by relations like *conjunctions* (and, or, etc.) and constructions like *although*, *since*, etc. Although simple sentences can express more than one event, the events other than the main one fill the grammatical roles of the main event. In complex sentences, events are not related through grammatical roles, but in structural relations instead. Simple sentences of Turkish are represented by the frame shown in Figure 3.3.

$$
\left[\begin{array}{ll}
\textit{clause-type} & \textit{predicative/attributive/existential} \\
\textit{s-form} & \textit{infinitive/adverbial/participle/finite} \\
\textit{voice} & \textit{active/reflexive/reciprocal/passive/causative} \\
\textit{speech-act} & \textit{imperative/optative/necessiative/} \\
 & \textit{wish/interrogative/declarative} \\
\textit{question} & \left[\begin{array}{ll} \textit{type} & \textit{yes-no/wh} \\ \textit{const} & \textit{list\_of(subject, dir-object, etc.)} \end{array}\right] \\
\textit{verb} & \left[\begin{array}{ll} \textit{root} & \textit{verb} \\ \textit{sense} & \textit{negative/positive} \\ \textit{tense} & \textit{present/past/future} \\ \textit{aspect} & \textit{progressive/habitual/etc.} \\ \textit{modality} & \textit{potentiality} \end{array}\right] \\
\textit{arguments} & \left[\begin{array}{ll} \textit{subject} & \textit{noun-phrase/sentential-clause} \\ \textit{dir-object} & \textit{noun-phrase/sentential-clause} \\ \textit{pred-property} & \textit{noun-phrase/sentential-clause} \\ \textit{source} & \textit{noun-phrase/sentential-clause} \\ \textit{goal} & \textit{noun-phrase/sentential-clause} \\ \textit{location} & \textit{noun-phrase/sentential-clause} \\ \textit{beneficiary} & \textit{noun-phrase/sentential-clause} \\ \textit{instrument} & \textit{noun-phrase/sentential-clause} \\ \textit{value} & \textit{noun-phrase/sentential-clause} \end{array}\right] \\
\textit{adjuncts} & \left[\begin{array}{ll} \textit{time} & \textit{noun-phrase/sentential-clause} \\ \textit{place} & \textit{noun-phrase/sentential-clause} \\ \textit{manner} & \textit{noun-phrase/sentential-clause} \\ \textit{path} & \textit{noun-phrase/sentential-clause} \\ \textit{duration} & \textit{noun-phrase/sentential-clause} \end{array}\right] \\
\textit{control} & \left[\begin{array}{ll} \textit{topic} & \textit{constituent} \\ \textit{focus} & \textit{constituent} \\ \textit{background} & \textit{constituent} \end{array}\right]
\end{array}\right]
$$

Figure 3.3: Representation of Turkish Simple Sentences

There is also a structural classification in the analysis of simple sentences and it is provided in *clause-type* feature. Predicative sentences are used to encode events defined in the ontology. The grammatical $verb$ is a lexeme whose category is verb in this type of sentences. They can take all the arguments shown in Figure 3.3 except *pred-property*. "Ali okula gitti" ("Ali went to the school") is an example of a predicative sentence. Attributive sentences corresponds to the entities with a set their properties defined in the ontology. The grammatical $verb$ is a lexeme whose category is either noun or adjective in attributive sentences. The structure of attributive sentences is simpler compared with predicative sentences, only $subject$ and *pred-property* from $arguments$, and $time$ and $place$ from $adjuncts$ are the allowed arguments of an attributive sentence. "Ali çalışkan bir öğrencidir" ("Ali is a hardworking student") and "Bu koltuk çok rahat" ("This armchair is very comfortable") are examples of attributive sentences. Existential sentences cover expressions of type "There is .../There are ..." and correspond to the sets which are the scopes of the speech-acts in TMR. They have the simplest structure with only the slots $subject$, $time$, and $place$. "Dün bölümde bir seminer vardı" ("There was a seminar at the department yesterday") is an example of existential sentence.

The feature $s\text{-}form$ is introduced to differentiate between normal sentences and sentential clauses which act as noun phrases to rich definitions of constituents. Normal sentences, defined in the previous paragraph, are represented by using $s\text{-}form/finite$ pairs. Sentential clauses that define acts like "to play football" in "Ali *top oynamak* istiyor" ("Ali wants to play football") are represented by $s\text{-}form/infinite$ pair. In this example, the sentential clause is the *dir-object* constituent of the main sentence "Ali wants to ...". Sentential clauses can be used as adjectives like "The child who was playing football" in sentence "*Top oynayan çocuk* camı kırdı" ("The child who was playing football broke the window") and this usage is represented by $s\text{-}form/participle$. The last pair, $s\text{-}form/adverbial$, covers sentential clauses which are used as adverbs like "by walking" in sentence "Okula *yürüyerek* gittim" ("I went to the school by walking").

Question slot is introduced to cover interrogative sentences. The pair *type/yes-no* is used to represent expressions like "Did you .../Is he ..." and $type/wh$ pair covers expressions like "Which book .../Who broke ...". The $const$ feature takes the thematic role as its value which is missing and queried in the sentence when its type is wh. So, in "Which book have you chosen?" the $const$ takes the value of $theme$ and in "Who broke the window?" $agent$ is the value of the feature $const$.

Control slot is introduced to handle free word ordering in Turkish. The feature *topic* defines the constituent which connects the sentence to the previous context and appears as the first constituent in the sentence. The feature *focus* is used for the constituent that is emphasized (important) and appears in the preverbal position. The last feature *background* defines the constituent that gives additional (but not necessary) information and appears in the postverbal position. So, the sentence "Pencereyi Ali kırdı dün" ("It was Ali who broke the window yesterday") has the *control* structure shown in Figure 3.4.

$$\begin{bmatrix} topic & dir\text{-}object \\ focus & subject \\ background & time \end{bmatrix}$$

Figure 3.4: An Example for Control Information

Complex sentences are constructed through combining simple sentences by conjunctions (and/or/etc.) and relations (since/although/etc), which are generally represented by domain relations in text meaning representation. Two new frames, shown in Figure 3.5, are introduced to cover complex sentences.

$$\begin{bmatrix} type & conjunction \\ conj & and/or/etc. \\ arg_1 & complex\text{-}sentence_1 \\ arg_2 & complex\text{-}sentence_2 \end{bmatrix}$$

$$\begin{bmatrix} type & linked \\ link\text{-}relation & rel\text{-}type \\ arg_1 & complex\text{-}sentence_1 \\ arg_2 & complex\text{-}sentence_2 \end{bmatrix}$$

Figure 3.5: Representation of Turkish Complex Sentences

The first frame in Figure 3.5, *conjunction*, is used for expressions like "Ali kitaplarını aldı ve okula gitti" ("Ali took his books and went to the school"). This sentence is represented by the f-structure shown in Figure 3.6.

The second frame, *linked*, is used when there is a relation between the two sentences like "Ali yeterince çalışmadığı için sınavı geçemedi" ("Since Ali didn't study enough, he couldn't pass the exam"). The f-structure corresponding to this sentence is given in Figure 3.7.

$$\begin{bmatrix} type & conjunction \\ conj & and \\ arg_1 & f\text{-}structure(\text{“}Ali\ kitaplarını\ aldı\text{”}) \\ arg_2 & f\text{-}structure(\text{“}Ali\ okula\ gitti\text{”}) \end{bmatrix}$$

Figure 3.6: An Example for Conjunctive Complex Sentences

$$\begin{bmatrix} type & linked \\ link\text{-}relation & için \\ arg_1 & f\text{-}structure(\text{“}Ali\ yeterince\ çalışmadı\text{”}) \\ arg_2 & f\text{-}structure(\text{“}Ali\ sınavı\ geçmedi\text{”}) \end{bmatrix}$$

Figure 3.7: An Example for Linked Complex Sentences

Noun phrases are the basic grammatical constructs that are used as the arguments of the verbal phrase (denotes the main event) in a sentence. So, arguments like *subject* and *direct-object* are generally noun phrases (only exception is the sentential clauses). Noun phrases of Turkish can be analyzed by dividing their structures into five main constructs [11, 28] and f-structure representation of noun phrases is shown in Figure 3.8:

- *Referent*: Contains the head of a noun phrase, which is the only mandatory element. It provides information about the word used as the head (its root and category) and its agreement properties (person,number). The simplest noun phrases like “adam” (“man”) are represented by just filling this slot.
- *Classifier*: Contains the constituents that classify the head noun with known entity sets such as “dış işleri bakanı” (“minister of foreign affairs”) and “fizik kitabı” (“physics book”).
- *Modifier*: Contains the constituents that give additional information about the head noun and they are analyzed in four categories:
  - *Modifying Relation*: Provides information about the properties, which can be a comparison with other entities, of the head noun, such as “vazo gibi bardak” (“glass like a vase”), “camdan kitaplık” (“book-case made of glass”).
  - *Ordinal*: Denotes the order of the head noun in a sequence of entities, such as “birinci koşucu” (“The first runner”), “son kitap” (“the last book”).

- *Quantative Modifier*: Expresses the quantity of the head noun in three different ways: by a *cardinal* like "üç kitap" ("three books"), with a *range* like "üç beş kitap" ("three to five books"), or with *fuzzy* adjectives like "çok fazla gürültü" ("too much noise").
- *Qualitive Modifier*: Gives qualitive properties of the head noun, such as "kırmızı kurşun kalem" ("red pencil"), "şişman çocuk" ("fat boy/girl").

$$
\begin{bmatrix}
\textit{referent} & \begin{bmatrix}
\textit{argument} & \begin{bmatrix} \textit{root} & \textit{lexeme} \\ \textit{category} & \textit{noun/verb/...} \end{bmatrix} \\
\textit{agreement} & \begin{bmatrix} \textit{person} & \textit{first/second/third} \\ \textit{number} & \textit{singular/plural} \end{bmatrix} \\
\textit{control} & \begin{bmatrix} \textit{drop} & \textit{positive/negative} \end{bmatrix}
\end{bmatrix} \\
\textit{classifier} & \textit{noun-phrase} \\
\textit{roles} & \begin{bmatrix} \textit{role} & \textit{agent/patient/theme/etc.} \\ \textit{arg} & \textit{case-frame} \end{bmatrix} \\
\textit{modifier} & \begin{bmatrix}
\textit{modifier-relation} & \begin{bmatrix} \textit{relation} & \textit{gibi/kadar/etc.} \\ \textit{argument} & \textit{list-of(noun-phrase)} \end{bmatrix} \\
\textit{ordinal} & \begin{bmatrix} \textit{order} & \textit{ilk/sonuncu/etc.} \\ \textit{intensifier} & \textit{positive/negative} \end{bmatrix} \\
\textit{quantifier-modifier} & \begin{bmatrix} \textit{low} & \textit{integer} \\ \textit{high} & \textit{integer} \end{bmatrix} \\
\textit{qualitive} & \begin{bmatrix} \textit{p-name} & \textit{adjective} \\ \textit{intensifier} & \textit{çok/en/sonuncu/etc.} \end{bmatrix} \\
\textit{control} & \begin{bmatrix} \textit{emphasis} & \textit{quantative/qualitive} \end{bmatrix}
\end{bmatrix} \\
\textit{specifier} & \begin{bmatrix}
\textit{set-specifier} & \textit{list-of(noun-phrases)} \\
\textit{specifier-relation} & \begin{bmatrix} \textit{relation} & \textit{dair/ait/etc.} \\ \textit{argument} & \textit{list-of(noun-phrase)} \end{bmatrix} \\
\textit{demonstrative} & \textit{bu/şu/o} \\
\textit{quantifier} & \begin{bmatrix} \textit{quantifier} & \textit{positive/negative} \\ \textit{definite} & \textit{positive/negative} \\ \textit{referential} & \textit{positive/negative} \\ \textit{specific} & \textit{positive/negative} \end{bmatrix}
\end{bmatrix} \\
\textit{possessor} & \begin{bmatrix}
\textit{argument} & \textit{list-of(noun-phrase)} \\
\textit{control} & \begin{bmatrix} \textit{drop} & \textit{positive/negative} \\ \textit{move} & \textit{positive/negative} \end{bmatrix}
\end{bmatrix}
\end{bmatrix}
$$

Figure 3.8: Representation of Turkish Noun Phrases

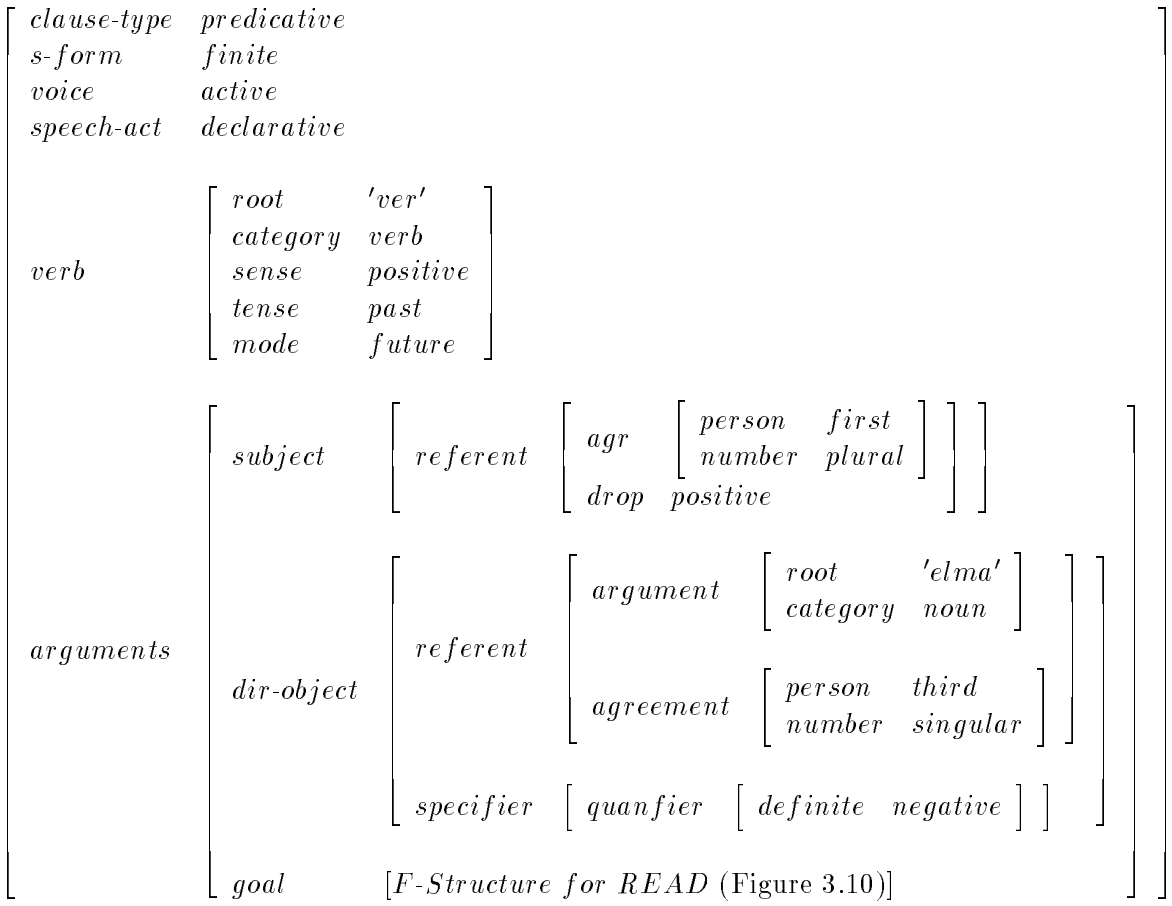

Figure 3.9: F-Structure of "Bir elma verecektik"

- *Specifier*: Contains the constituents that are used to make a distinction between the head noun and a set of similar nouns in the context. These constituents are also divided into five categories:
  - *Quantifier*: Denotes the the quantity of head noun, such as "her çocuk" ("every child"), "bazı kitaplar" ("some books").
  - *Demonstrative*: Used to point out the head noun, such as "bu kitap" ("this book"), "şu çocuk" ("that child").
  - *Specifying Relation*: Used to distinguish the head noun through mentioning its relationship with other entities, such as "kitabın solundaki kalem" ("pencil at the left of the book").
  - *Set Specifier*: Used to express head nouns that are members of a specific set, such as "kalemlerden kırmızısı" ("The red one among the pencils").

- *Possessor*: Denotes the owner of the head noun, such as "onun kalemi" ("his pencil"), "çocuğun kitabı" (the child's book), etc.

3.3.1 An F-Structure Example

To show how a sentence in Turkish is represented by using f-structure representation, the same example which is used for the explanation of TMR construction is given. The corresponding f-structure representation of the sentence below is given in Figure 3.9 and Figure 3.10

"Kitap okuyan kadına bir elma verecektik"
("We were going to give an apple to the woman who was reading a book")

$$
\left[\begin{array}{ll}
clause\text{-}type & predicative \\
s\text{-}form & participle \\
voice & active \\
speech\text{-}act & declarative \\
\\
verb & \left[\begin{array}{ll} root & 'oku' \\ category & verb \\ sense & positive \\ tense & past \end{array}\right] \\
\\
arguments & \left[\begin{array}{ll}
subject & \left[\begin{array}{ll} referent & \left[\begin{array}{ll} arguments & \left[\begin{array}{ll} root & 'kadın' \\ category & noun \end{array}\right] \\ agreement & \left[\begin{array}{ll} person & third \\ number & singular \end{array}\right] \end{array}\right] \end{array}\right] \\
dir\text{-}object & \left[\begin{array}{ll} referent & \left[\begin{array}{ll} arguments & \left[\begin{array}{ll} root & 'kitab' \\ category & noun \end{array}\right] \\ agreement & \left[\begin{array}{ll} person & third \\ number & singular \end{array}\right] \end{array}\right] \end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Figure 3.10: F-Structure of "Kitap okuyan kadın"

3.4 Generation Map-Rules

There is not any knowledge about a specific natural language in text meaning representation. So, information such as grammatical roles and syntactic properties of a sentence is not available in the frames of a TMR. But, such information should be used in tactical generation for handling a language's syntactic encoding of meaning. The interface between f-structure representation of a language's sentence and TMR is achieved by using map-rules. Map-rules encode language specific information about how meaning resides in TMR is related to the target language's syntactic structure. A map-rule is used to check the contents of TMR frames for finding specific information, and update the syntactic properties of the current sentence if this information is found in the input TMR [10, 22, 20].

Map-rules are developed for the following purposes [10]:

- To relate thematic roles, such as *agent*, *theme*, *source*, of events to their corresponding grammatical roles, such as *subject*, $dir-object$, in target sentences.
- To create specific features in the f-structure. Some examples of such features are *tense*, *clause-type*, *number*, and *person*. Their values are determined by checking the existence of various $filler/value$ pairs in a set of TMR frames.
- To find the relations between the events of a single sentence. These relations are extracted from either domain relations or available contextual information. Contextual relatedness can be explained by the sentence "John, who came to your birthday party, went to America last month". In this sentence, the event "came to your birthday party" is used as a definite description for John.
- To update an information which was created previously by a more general map-rule and should be changed to handle new information extracted from TMR. The passivization rule which changes the verb's argument structure is an example of this type of map-rules.
- To create a new slot in the f-structure whose value is not directly mapped from a semantic slot in TMR. This type of map-rules are generally created to introduce closed-class lexical items to the f-structure such as prepositions, conjunctions, etc.

The structure of a map-rule is the following [10, 20]:

Map-Rule
Generation-Language
Rule-Type
Application-Type
Content-Conditions
New-Information

The first slot, *Generation-Language*, provides the name of the language for which that map-rule is written for. The second slot, *Rule-Type*, denotes the type of the entity that map-rule is created for. The value of *Rule-Type* can be a lexical item from the lexicon, a concept from the ontology, the name of a TMR frame which is not an instantiated concept (modality, speech-act, etc.), or the name of some special language phenomena (like one way of relating events described in Chapter 4). A map-rule whose *Rule-Type* is a lexical item is generally created for two reasons: to provide the relation between the thematic structure of that lexical item and its corresponding grammatical role structure (kir_1 in Figure 3.11, and to introduce closed-class words that should be used with that lexical item in certain contexes.

TMR frames which are instantiations of concepts are processed by map-rules written for those concepts. Such map-rules cover general syntactic properties of those concepts, and they are designed in such a way which follows the hierarchical structure of the ontology. That is, map-rules that can be applied to all of the children of a concept is attached to that parent concept. In this way, enumeration of common map-rules is avoided. So, the set of applicable map-rules of a concept consists of all map-rules created for its ancestor concepts and that concept itself.

For example, in Figure 3.11, starting with the concept *BREAK*, the *PUNCTUALITY* concept determines the tense of the sentence, *EVENT* concept creates the *clause-type* and the *sense* information, and the *ALL* concept introduces *sentence-form* feature. So, map-rules which are applicable to a TMR frame that is an instantiation of a concept can be collected by just starting with that concept and traversing the ontology in a bottom-up fashion until the root concept *ALL* is reached.

Map-rules whose *Role-Type* are the names of special TMR frames (focus, attitude, domain relations, etc.) are created for processing semantic and pragmatic phenomena that are introduced to meaning representation with those frames. The inheritance mechanism used for processing concepts is not applicable to map-rules designed for these special frames. These map-rules are used to process information contained in those special frames, and a map-rule created

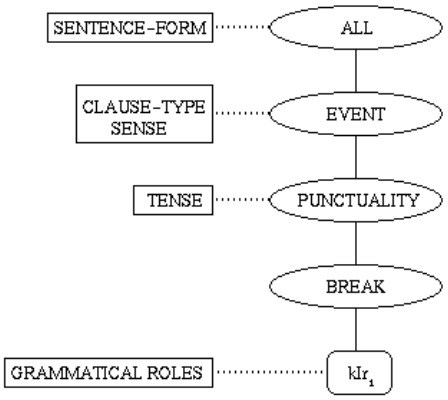

Figure 3.11: An Imaginary Map-Rules Structure

for such a frame cannot be used for other frames. For example, map-rules can be developed to determine *speech-type* and *voice* features of a f-structure from *speech-act* frame in a TMR.

The third slot, *Application-Type*, in a map-rule can be filled with two different values, *exclusive* or *any*, and its value determines the processing type of that map-rule. The first type, *exclusive*, is utilized to create a set of mutually exclusive structural mappings which are used to determine the value of a feature from a set of TMR frames Determination of the *tense* of a sentence and determination of noun phrases' agreement values (*number*, *person*) are examples which require exclusive processing. Map-rules of type *exclusive* are designed such that their conditions for success are contradictory. That is, only the conditions of one rule from that set can succeed in any context which can exist in any TMR. For example, in determination of *tense* value, contents of an event's aspect frame, its temporal relation with speech moment, and any modalities that are available in the input TMR are checked by those exclusive rules, and only one rule succeeds with a returned *tense* value. To design map-rules which are mutually independent from others, the second method of application, *any*, is provided. This method is used for separating map-rules which are created for introducing different syntactic phenomena of the target language. So, the general structure of map-rules associated with a single entity is like in Figure 3.12.

The fourth slot, *Content-Conditions*, specifies the meaning requirements that must be satisfied for the application of a map-rule. These requirements are represented as a list of references to TMR frames and their contents. To apply a map-rule, each reference must be found in TMR. Since the content of a TMR is not limited and predetermined, map-rules developed for a language should be TMR independent. So, making references to arbitrary frames and their contents

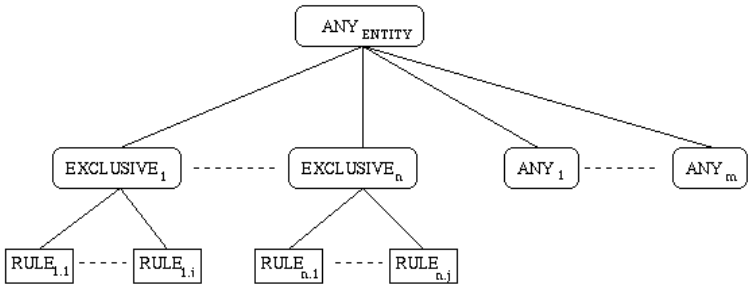

Figure 3.12: Map-Rules Structure of an Entity

are meaningless, and only four frames and their contents are allowed to be referred in defining the meaning requirements of a map-rule. These four frames are:

- $Frame_{processing}$ for which the map-rule is activated
- $Frame_{event}$ which is the event that contains $Frame_{processing}$
- $Frame_{context}$ is utilized for handling special events that provide nonpropositional meaning (like modality and speech-act) to meaning representation. Event $HEAR$ (in "I heard that ...", the usage of $HEAR$ is not an event, it is for denoting *epistemic* modality) is an example of $Frame_{context}$ usage.
- $Frame_{speech}$ which is the speech-act frame of the current TMR.

The contents of these frames can also be referenced. Also, frames which are used to introduce linguistic phenomenon for these frames such as aspect and modality frames can be used in specifying meaning requirements.

There are various methods provided for checking the content of TMR frames. The first method is used whenever a special $feature/value$ pair is needed, and the $feature/value$ pairs of a linguistic frame are tested by special rules. For example, the first example given below checks whether the *polarity* feature of $Frame_{processing}$ is *positive* or not, and the second one tests the whole *aspect* frame of $Frame_{event}$ for values $(perfect, single, momentary, false)$.

$$ref(Frame_{processing}, polarity, positive)$$
$$aspect(Frame_{event}, [perfect, single, momentary, false])$$

The second method is utilized when the existence of a feature is the only criterion that is required. For applying this method, two rules are provided: *exist* and *not-exist*, and their structures are given in the following examples (note that, the rule *exist* returns the value of *agent* feature if it is inside the $Frame_{processing}$, and the rule *not-exist* succeeds if $Frame_{speech}$ does not contain *focus* feature).

$exist(Frame_{processing}, agent, Value)$
$not\text{-}exist(Frame_{speech}, focus)$

If there is a requirement to check whether different references are pointing to same TMR frame or not, or whether two different frames are coreferenced or not in the input TMR, the third method is utilized. Four rules are provided to handle this type of constraints: *same*, *not-same*, $coref$, and *not-coref*, and their usage are shown by the following examples:

$same(Frame_{processing}, Frame_{event})$
$coref(speaker, Frame_{processing})$

To check whether a set of frames are related to each other through frames like temporal relation or domain relation, the fourth method is used. It gets a set of frames and the relation type between them, and searches in TMR whether they are all connected through that relation type or not. To exemplify this usage, the following examples are given:

$time(after, [Frame_{speech}, Frame_{event}, Frame_{context}])$
$domain(reason, [Frame_{context}, Frame_{event}])$

The fifth slot, $New\text{-}Information$, specifies the update operations which will be performed by that map-rule on the f-structure. If all the requirements specified in *Content-Conditions* are satisfied, then a list of new information are processed to update the f-structure being constructed. Three types of update operations are provided: $feature\text{-}addition$, *slot-addition*, and *slot-to-slot-mapping*. $Feature\text{-}addition$ adds a new $feature/value$ pair to the f-structure, *slot-addition* inserts a new slot in the f-structure which is not created yet, and *slot-to-slot-mapping* bounds the features that are created for a TMR frame to a slot in the f-structure. Their structures are given by the following examples:

$feature(tense, past)$
$slot(dir\text{-}object)$
$map(Frame_{event}.agent, subject)$

The overall structure of map-rules is exemplified with the following examples:

$maprule(turkish, kır_1, exclusive,$
$[exist(Frame_{processing}, agent, Slot_1),$
$exist(Frame_{processing}, patient, Slot_2),$
$not\text{-}exist(Frame_{speech}, focus)]$
$[map(Slot_1, subject), map(Slot_2, dir\text{-}object)])$

$$maprule(turkish, punctuality, exclusive,$$
$$[modality(Frame_{event}, [deontic, equal(1), speaker, Frame_{event}]),$$
$$modality(Frame_{speech}, [epistemic, equal(0), speaker, Frame_{event}]),$$
$$aspect(Frame_{event}, [perfect, single, momentary, false]),$$
$$time(after, [Frame_{speech}, Frame_{event}])]$$
$$[feature(desc\text{-}verb, necessiative), feature(mode, past)])$$

3.5 Generation Lexicon

Lexicon is one of the two knowledge resources which are utilized to establish the connection between meaning representation and the target language. It contains information about open-class word senses and closed-class lexemes of the target language which can be utilized in any phase of the generation. Each entry in the lexicon corresponds to a word sense of the target language and provides information about word's phonological, morphological, syntactic, semantic, and pragmatic properties [25, 10, 20]. Such information can be used in the selection of words to be used in target sentences, the introduction of syntactic realizations, and the resolution of inflectional and sound changes in the final construction [7, 10, 14].

Each entry in the lexicon consists of a number of slots (each possibly having multiple fields) for integrating various levels of lexical information, and that entry is indexed on its sense for a specific word. The slots which can be used in the definition of a lexeme are CAT (syntactic category, such as $verb$, $noun$), $ORTH$ (orthography, eg. abbreviations, variants), $PHON$ (phonological knowledge), $MORPH$ (irregular forms, inflectional properties, inflection classes), SYN (syntactic features, such as countability), $STRUC$ (sentence or phrase-level syntactic inter-dependencies), SEM (semantic information, such as subcategorization,its parent concept from the ontology, and its meaning definition), and $PRAGM$ (pragmatic knowledge, such as stylistic information). To exemplify how a word sense is defined in the generation lexicon, the following examples are given.

$adam_1$			$kır_1$		
CAT	*category*	*noun*	CAT	*category*	*verb*
	root	*adam*		*root*	*kır*
SYN	*countable*	*yes*	SEM	*is-a*	$BREAK$
	proper	*no*		*subcat-info*	
SEM	*is-a*	$HUMAN$		*requires*	*patient*
	definition			*optional*	*agent*,
	type	*common*			*means*
	gender	*male*			
	age	≥ 17			

Five of the slots used in the definition of a lexeme, which are $ORTH$, $PHON$, $MORPH$, SYN, and $STRUC$, are utilized in tactical generation. Since this phase is out of our scope, these slots are not explained in detail. The interface between concepts (denoting events and entities) used in a TMR and words of the target language is established using semantic and pragmatic properties of words provided in the lexicon. Each entry whose category is noun or verb is an instantiation of a concept from the ontology, and this information is given in SEM slot with *is-a* link. So, for every TMR frame which is an instantiated concept, there is a set of candidate lexicon entries which are also children of the same concept. Since there can exist more than one entry in the lexicon for a concept, it is important to choose the most appropriate one. This selection problem is overcome by using various sources of information that is provided in SEM and $PRAGM$ slots.

Since a verb cannot take all thematic roles defined in Chapter 2 as its argument, its thematic structure should be constrained. Also, some verbs cannot be used without certain thematic roles. All such information is provided in SEM slot under subcategorization feature. For example, word 'kır' in Turkish, which is corresponding to word 'break' in English, cannot take the thematic roles *source* and *destination*, and it should be used at least with a *patient*. It can also take roles *agent* and *means*, but they are optional, eg. 'Cam kırılmış.' ('The window was broken.'). This information is given in $subcat\text{-}info$ which contains the list of roles that a lexeme requires and takes optionally. The rest of the roles are assumed to be rejected by that lexeme. The following is the structure of subcategorization information:

$$
\begin{array}{lll}
lexeme_i & & \\
\quad SEM & subcat\text{-}info & \\
 & \quad requires & list\text{-}of(thematic\text{-}roles) \\
 & \quad optional & list\text{-}of(thematic\text{-}roles)
\end{array}
$$

The second source of information is also provided to limit the thematic structure of an event. Although the subcategorization information supplies the general thematic structure, it has no constraint on the values of these thematic roles. The values of thematic roles can also be restricted to specific concepts from the ontology. These cases generally captures word senses which have very specific usages. For example, the verb 'look up' can be a child of concept $SEARCH$, but its *theme* should be something like a textual source of knowledge. To handle such phenomenon, there is a slot $role\text{-}value\text{-}info$ in SEM slot which introduces

such restrictions on the values of thematic roles. The following is the structure of role value information:

$$
\begin{array}{lll}
lexeme_i & & \\
\quad SEM & role\text{-}value\text{-}info & \\
 & \quad role_1 & list\text{-}of(allowed\text{-}concepts) \\
 & \quad role_2 & list\text{-}of(allowed\text{-}concepts) \\
 & \quad \vdots & \quad \vdots
\end{array}
$$

So, the example 'look up' has the following lexicon entry:

$$
\begin{array}{lll}
look\text{-}up_1 & & \\
\quad SEM & is\text{-}a & SEARCH \\
 & subcat\text{-}info & \\
 & \quad requires & agent, theme \\
 & \quad optional & manner \\
 & role\text{-}value\text{-}info & \\
 & \quad theme & DICTIONARY/ENCYCLOPEDIA
\end{array}
$$

The most important source of information on which the selection task depends is the meaning definitions of lexicon entries. The definition of a lexeme is achieved by constraining the meaning space of the parent concept. The meaning space is limited by reducing the size of the value domains that concept is defined on. For example, the concept $HUMAN$ corresponds to all of the words of a language which are used for referring to a human-being. But usages of these words are limited by the properties of the human-being that is referred to (eg. the word 'man' cannot be used for a human whose gender is female). The meaning definition is also contained in SEM slot with $definition$ slot. So, the definition of word 'car' can be the following:

$$
\begin{array}{lll}
car_1 & & \\
\quad SEM & is\text{-}a & VEHICLE \\
 & definition & \\
 & \quad power & motor \\
 & \quad surface & road \\
 & \quad wheels & four\text{-}wheels \\
 & \quad purpose & human\text{-}transportation
\end{array}
$$

The last source of information which is provided in the lexicon is about pragmatic properties of lexemes. Especially, the stylistic knowledge affects word selection. For example, the words chosen in formal writing, literature, and speech between close friends are quite different. The usage of slang words is a common practice in daily speaking between friends, which is very inappropriate

in formal situations. To handle such usages, stylistic constraints are provided under *stylistics* in $PRAGM$. For example, the word 'herif', which has same meaning as 'adam' ('man' in English), in Turkish is used in informal situations with a negative meaning. So, its $PRAGM$ slot can be the following:

$herif_1$			
	SEM	*is-a*	$HUMAN$
		definition	
		type	*common*
		gender	*male*
		age	≥ 17
	$PRAGM$	*stylistics*	
		formality	0
		respect	0
		politeness	0
		simplicity	1
		attitude	
		type	*evaluative*
		value	0.2

In addition to information about meaning and usage differences between words, the lexicon also provides knowledge about the relation between the thematic roles of a lexeme with its grammatical realizations. That is, map-rules specific to a lexical item may be introduced in the lexicon. This information is utilized whenever there is an ambiguity about the mapping between thematic roles and their grammatical counterparts. For example, noun phrases are generally the fillers of both *subject* and *dir-object*. So, whether the f-structure created for *agent* is placed inside *subject* or *dir-object* slot is determined by using map-rules provided by the lexical item. This knowledge is defined in *map-rules* of SEM slot. The following can be an example to clarify the usage of this information:

$kır_1$
SEM *is-a* $BREAK$
map-rules
$map\text{-}rule(turkish, kır_1, exclusive,$
$[exist(Frame_{processing}, agent, Slot_1),$
$exist(Frame_{processing}, patient, Slot_2),$
$not\text{-}exist(Frame_{speech}, focus)],$
$[map(Slot_1, subject),$
$map(Slot_2, dir\text{-}object)])$

Chapter 4

Computational Model

The computational model described in this chapter is designed to transfer the TMR of a sentence to its corresponding feature structure representation of that sentence in the target language. To achieve this task, the model should select the open-class words to be used in that sentence, construct its syntactic structure, determine the grammatical roles, and introduce closed-class lexical items through processing the frames in the TMR [10, 22].

The model developed is language independent, that is, in its processing modules there is no special information about a target language. The relation between the abstract representation (TMR) of a sentence and the f-structure representation of that sentence in the target language is constructed by using separate knowledge resources developed for the target language. These knowledge resources, explained in Chapter 3, are the *lexicon* (word information), the *map-rules* (the relation between the information in TMR and f-structure of the sentence), and the feature structure representation of the target language (the encoding of syntactic structure). So, to produce the f-structure of a sentence in a target language from an input TMR, these three knowledge sources should be developed and introduced to the computational model as the knowledge resources of the target language. Turkish is chosen as a target language to test the developed computational model. To achieve generation, small-sized resources of Turkish lexicon and map-rules together with a complete Turkish f-structure representation are provided.

The model processes the frames in the TMR one by one in a specific order. This order is dynamically updated depending on the obtained information from the frame being processed. There are two types of frame-processing operation, and one of them is selected depending on whether the processed frame is an instantiated concept or not. The frames that are instantiated concepts should

introduce open-class lexical items, and all the map-rules that are derived from the ontology together with the map-rules associated with the selected lexical item should be processed. The other frames, which are used to introduce semantic and pragmatic information in a TMR, at most select closed-class lexemes and only the map-rules associated with their type are processed [10, 22].

These two tasks, lexical selection and map-rule application, are handled in two separate submodules. Although these two submodules work in parallel in the main module, their way of processing TMR can be developed separately. The first submodule, lexical selection module, is activated for the frames that are instantiations of concepts. It uses the lexicon developed for the target language to decide the open-class word to be used for that concept in the target sentence. First, it creates a candidate lexeme set by using the relation between the concepts and lexemes, and selects the most appropriate one from that set by using the meaning in TMR and properties of lexemes. The second one, map-rule application module, processes all map-rules associated with the current frame and updates the constructed f-structure. It first collects the set of applicable map-rules using the lexicon, the ontology, and the map-rules knowledge base, and then fires them in the order in which they are collected. First, it checks whether the conditions required for the application of a map-rule are satisfied by the information in the TMR or not, and it updates the f-structure for the successful map-rules by using the feature structure representation of the target language. The architecture of the computational model developed in this work is described in Figure 4.1.

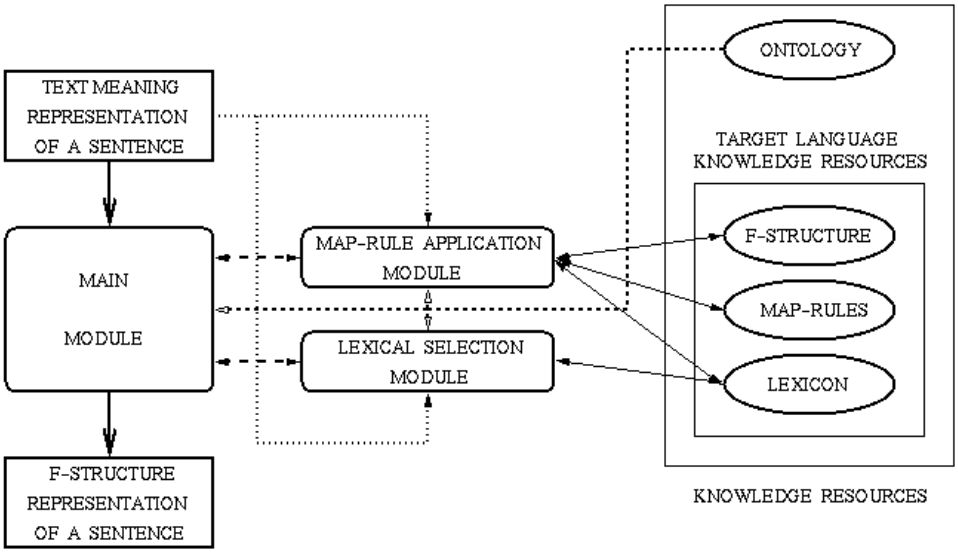

Figure 4.1: Computational Model

Each submodule and their usage in the main model are explained in the

following three subsections: the lexical selection module, the map-rule application module, and the main module. Finally, how the computational model works on a TMR is exemplified through a simple Turkish sentence.

4.1 Lexical Selection Module

The task of selecting the most appropriate words for the target language sentence is handled in this module. There are two main problems to be solved for lexical selection in generation: resolution of synonymy and near-synonymy, and imperfect matches between the meaning in a TMR and the words of a target language. The first problem can be defined as selecting between a set of lexemes which introduce nearly the same meaning in the target language. The second problem is encountered when a word in the source language, eg. English, has not any matching word in the target language, eg. Turkish. This problem arises since sometimes the source language makes finer differentiation on an event or an entity through different wording. In order to resolve these problems, knowledge available in the lexicon is utilized such as stylistic information, subcategorization information, meaning definition [7, 22].

In order to achieve the goal of selecting near-perfect open-class words, the lexical selection module should carry the meaning resides in the TMR frame into the target sentence. So, the lexical selection in this work is mainly based on the meaning distance between the TMR frame and the lexemes in the candidate set. To do this, a distance assigning capability between the meaning of the TMR frame and the meaning introduced by the use of the candidate word in the target language is utilized. This module calculates proximities between the meaning in a TMR and each candidate lexeme, and returns the closest one as the selected lexeme. In calculation, the module makes use of the definition of a lexical item provided in the lexicon. Although the proximity of meaning is the major criterion used in the lexical selection, there are cases in which meaning comparison is not enough for perfect selection. In such cases, the semantic structure of a lexical item with its pragmatic properties should also be taken into account. Such information is also obtained from the lexicon.

The lexical selection module can be divided into two distinct phases: context-dependent selection and context-independent selection. The first phase checks the semantic structure constraints of the candidates and eliminates those whose requirements are not satisfied by the TMR. The second phase sorts the remaining candidates according to their calculated proximities to the meaning in

the TMR frame and returns the one with the minimum distance. If there is still ambiguity in the selection, pragmatic constraints such as stylistic information is utilized in this phase and finally the best-matched candidate is returned as the selected word [10]. In the following subsections, the context-dependent, the context-independent selection, and the main algorithm are analyzed in detail.

4.1.1 Context-Dependent Selection

The semantic structure of an event or an entity represented in a TMR can affect the choice of lexical item to be used in a language. This submodule checks whether the semantic structural constraints of a lexical item is satisfied in the TMR frame and the lexemes that reject the current structure are eliminated from the candidate list. Currently, two different sources of information are utilized in this module:

- *Subcategorization Requirements*
- *Role-Value Requirements*

Subcategorization requirements, as explained, identify the thematic structure of a word. Each word in the lexicon has information about the roles it requires and takes optionally. These requirements are compared with the thematic structure of the TMR frame for which the lexical item to be selected. If there is a mismatch between the requirements and the structure of that frame, then the candidate lexeme is rejected. There are two sources of mismatches: a role required for the lexeme is not available in the TMR frame, or a role inside the TMR frame is rejected by that lexeme. For example, consider the following two lexemes:

$$
\begin{array}{lll}
lexeme_i & & \\
\quad is\text{-}a & : & CONCEPT_m \\
\quad subcat\text{-}info & : & \\
\quad\quad requires & : & agent \\
\quad\quad optional & : & theme
\end{array}
\qquad\qquad
\begin{array}{lll}
lexeme_j & & \\
\quad is\text{-}a & : & CONCEPT_m \\
\quad subcat\text{-}info & : & \\
\quad\quad requires & : & agent, theme \\
\quad\quad optional & : & goal
\end{array}
$$

Since both lexemes are instances of $CONCEPT_m$, they are in the list of the lexeme candidates for a TMR frame which is an instantiation of that concept. If there is such a frame, then the following selectional criteria is used to check the thematic role constraints:

if TMR frame has role *agent*
then if TMR frame has role *theme*
then if TMR frame has role *goal*
then select $lexeme_j$
else select $lexeme_i$ & $lexeme_j$
else select $lexeme_i$
else reject both lexemes

Role-Value requirements, further limit the usage of a word through constraining the values which the thematic roles can take. This kind of limitation is introduced for lexemes with very specific usages. Generally such lexemes are introduced to define a specific instantiation of an event or an entity which reduce the number of words for expressing the same meaning, eg. terminological lexicals. Whenever such information is available in the lexicon, this module checks the values of those roles in the TMR frame and eliminates the lexemes that have mismatches. For the following example, having the thematic role *agent* is not enough for a TMR frame to satisfy semantic constraints of the $lexeme_k$, the value of its *agent* should be an instantiation of the concept $HUMAN$.

$lexeme_k$
is-a : $CONCEPT_i$
subcat-info :
requires : $AGENT$
optional : $PATIENT$
role-value-info :
$AGENT$: $HUMAN$

4.1.2 Context-Independent Selection

The meaning wanted to be expressed in a language affects the word choice since each word provides a specific range of meaning to the sentence in which it is used. This meaning contribution of a word is defined in the lexicon and it is compared with the meaning resides in the TMR frame to calculate its proximity. This module is responsible for calculating a penalty for each candidate lexeme, which corresponds to the proximity between the meaning in the TMR frame and the lexicon entry of that lexeme. The candidates are sorted with respect to those assigned penalties, and the lexeme with the minimum penalty is returned as the selected word for that frame. In proximity calculation, every slot-value pair in the lexeme and the TMR frame are compared, and in the cases of imperfect matches, penalty values are assigned to those slots. These penalties are normalized by the contribution ratio of those slots to the overall meaning in the usage of that lexeme.

The ratios are defined in the lexicon by importance values which are introduced whenever appropriate. The overall distance is computed through adding the individual penalties assigned to the slots that have different values. The following heuristics are utilized in this module for the calculation of the proximity.

1. Introduction of extraneous meaning is minimized by assigning a predefined maximum penalty to a slot which is used in definition of the lexeme, but not contained in the TMR frame.

2. Uncoverage of meaning is reduced by assigning a fraction of the maximum penalty to a slot which is contained in the TMR frame, but not used in the definition of the lexeme.

3. Meaning match of a slot, which exists in both the lexeme and the TMR frame, is proportional to the distance in ordered values such as color, and the size of intersection in values of range such as age.

4. The calculated match is normalized by the domain sizes of the feature-values to minimize the distance in larger domains compared with smaller ones.

5. The final distance returned by the fourth heuristic is rated by its importance on overall meaning, so a mismatch on a less important slot will have a smaller influence on the proximity.

6. The quality of total match is the sum of weighted penalties of common slot-filler pairs together with penalties from the first two heuristics.

In calculating the distance between the values of a common slot, two things should be taken into account: the domains (or value sets) that define the allowed values of a filler, and the cardinality of the filler in the meaning pattern. According to the relationship between the values, domains can be divided into three different types:

- *Unordered Domains* : When the values are symbolic and they cannot be ordered on a specific metric, their domain is declared to be *unordered*. Since there is no ordering between the values, the distance between any two values in such a domain is assumed to be equal. The domain of things that can be read (book, article, newspaper, etc.) is an example of an *unordered domain*.

- *Discretely-Ordered Domains* : If the values are symbolic and they are ordered according to some criteria, then their domain is declared to be *discretely-ordered*. Because of ordering, the distance between two values in such a domain is proportional to the number of values lies between them. Months of a year can be an example for a *discretely-ordered domain*.

- *Continiously-Ordered Domains* : When the values are numeric type and the standard order is used, their domain is declared to be *continiously-ordered*. In such domains, the distance between two values are proportional to the difference between them. This difference is normalized with the smallest unit of the domain increments. For this type of domains, height and age can be given as examples.

The distance metrics used in context-dependent selection for domain types in cases of value mismatches are given below:

TYPE	*DISTANCE*	*EXAMPLE*
Unordered	1	$value_{frame} = newspaper$, $value_{lexeme} = magazine$, $assigned\text{-}distance = 1$
Discretely Ordered	$num\text{-}of\text{-}values\text{-}between + 1$	$value_{frame} = february$, $value_{lexeme} = may$, $values\text{-}between = \{march, april\}$, $assigned\text{-}distance = 2 + 1 = 3$
Continiously Ordered	$\frac{abs(value_{tmr} - value_{lexeme})}{unit\text{-}size}$	$value_{frame} = 60$, $value_{lexeme} = 48$, $unit\text{-}size = 2$, $assigned\text{-}distance = \frac{60-48}{2} = 6$

It is mentioned that, in the fourth heuristic, the distance calculated is normalized by the size of the domain. Since there is a maximum penalty defined for the first heuristic, the distance should also be normalized by this maximum penalty. So, the following equation gives the final penalty for a slot with mismatched values:

$$penalty = maximum\text{-}penalty \times \frac{distance\text{-}calculated}{domain\text{-}size}$$

The final penalty of a slot also depends on the cardinality of values which are used as fillers in a TMR frame. The cardinality of a filler in a TMR frame can be one of the following three types:

- *Single Fillers* : If a filler in a TMR frame is filled with only one value, then its cardinality is defined as *single*. Because of one value, the distance is calculated by just taking into account the domain of the filler. Animate gender (exclusive) can be an example for *single filler*.

- *Enumerated Fillers* : When a filler in a TMR frame is filled with more than one value, its cardinality is defined as *enumerated*. In such a filler, every value of the set must be considered. So, the method utilized for such fillers is to compute the distance for every value and combine these penalties by some criterion. Currently, there are two methods for combining: disjunctive merging (the minimum of the penalties is assumed to be the overall penalty) and conjunctive merging (the mean of the penalties is calculated and assigned as the overall penalty). For example, the *wheels* slot in the definition of a motorbike, $[vehicle, [[power, motor], [wheels, [2, 3]]]]$, can be an example for *enumerated filler*.

- *Ranged Fillers* : When a filler is filled with a range values from an ordered domain, either discrete or continuous, its cardinality is defined as *ranged*. In such a filler, the size of the intersection between the filler of the TMR frame and the lexeme is the major criteria for calculating the proximity. But, also the range size of each filler is contributed to the final penalty to ensure that the small-sized ranges are preferred to larger ones when the intersection size is equal. An example for *ranged fillers* is the definition of a human set whose age is in a specific age range such as childhood $[human, [[type, common], [gender, unknown], [age, (4, 12)]]]$.

If the cardinality of filler is not *single*, the final penalty is calculated by the following equations:

TYPE	*PENALTY*	
Enumerated	$penalty_i$: assigned penalty for $value_i$ *in TMR frame*	
	n : the cardinality of the filler	
	$penalty = \min_{i=1}^{n} penalty_i$	$(Disjunctive)$
	$penalty = \frac{1}{n} \times \sum_{i=1}^{n} penalty_i$	$(Conjunctive)$
Ranged	$inter$: size of intersection	
	$range_1 :\ size\text{-}of\text{-}range_{TMR} - inter$	
	$range_2 :\ size\text{-}of\text{-}range_{lexeme} - inter$	
	$penalty = inter - \frac{1}{2} \times (range_1 + range_2)$	

After a penalty is calculated for every slot in both the TMR frame and the lexeme, total penalty is calculated by the following equation in which n is the

total number of different slots found and $importance_i$ defines the contribution ratio of $slot_i$ to overall meaning defined in the lexicon (assumed to be 1 if not defined).

$$total\text{-}penalty = \sum_{i=1}^{n} penalty_i \times importance_i$$

We can exemplify the distance calculation methods introduced in this section by the following example. Both the TMR frame, which is an instantiation of $concept_i$, and the $lexeme_j$, which inherits that concept, are imaginary to show all types of calculations that can be done.

$concept_i$		$lexeme_j$		
$slot_{c1}$	$value_{c1}$	*is-a*	$concept_i$	
$slot_2$	$value_{c2}$	*definition*		
$slot_3$	$value_{c3}$	$slot_{l1}$	$value_{l1}$	$(importance_1 = 0.6)$
$slot_4$	$value_{c4}$	$slot_2$	$value_{l2}$	$(importance_2 = 1.0)$
$slot_5$	$\{value_{c5}, value_{c6}\}$	$slot_3$	$value_{l3}$	$(importance_3 = 0.8)$
$slot_6$	$(value_{c7}, value_{c8})$	$slot_4$	$value_{l4}$	$(importance_4 = 0.2)$
		$slot_5$	$value_{l5}$	$(importance_5 = 0.4)$
		$slot_6$	$(value_{l6}, value_{l7})$	$(importance_6 = 0.6)$

$maximum\text{-}penalty = 10$
$ratio(for\ heuristic_2) = 0.8$

SLOT	*DOMAIN TYPE*	*PENALTY CALCULATIONS*
$slot_{l1}$	not important	$slot_{l1}$ is not in the frame, $heuristic_1$, $penalty_1 = max\text{-}penalty = 10$
$slot_{c1}$	not important	$slot_{c1}$ is not in the lexeme, $heuristic_2$, $penalty_2 = max\text{-}penalty \times ratio(for\ heuristic_2) = 8$
$slot_2$	*Unordered*	$domain\text{-}size = 4$, $distance_3(value_{c2}, value_{l2}) = 1$, $penalty_3 = 10 \times (1/4) = 2.5$
$slot_3$	*Discretely Ordered*	$distance_4 = values\text{-}between(value_{c3}, value_{l3}) = 3$, $domain\text{-}size = 10$, $penalty_4 = 10 \times ((3+1)/10) = 4$
$slot_4$	*Continiously Ordered*	$distance_5 = difference\text{-}between = value_{c4} - value_{l4} = 8$, $unit\text{-}size = 1$, $domain\text{-}size = 50$ $penalty_5 = 10 \times ((8/1)/50) = 1.6$
$slot_5$	*Discretely Ordered*	$domain\text{-}size = 5$, $distance_{6.1} = values\text{-}between(value_{c5}, value_{l5}) = 1$, $penalty_{6.1} = 10 \times ((1+1)/5) = 4$, $distance_{6.2} = values\text{-}between(value_{c6}, value_{l5}) = 0$, $penalty_{6.2} = 10 \times ((0+1)/5) = 2$, $penalty_{6(disjunctive)} = min(4, 2) = 2$, $penalty_{6(conjunctive)} = \frac{1}{2}(4+2) = 3$

SLOT	*DOMAIN TYPE*	*PENALTY CALCULATIONS*
$slot_6$	*Continiously Ordered*	$domain\text{-}size = 20,\ unit\text{-}size = 1,$ $intersection\text{-}size = 6 = inter,$ $range\text{-}size(frame) = 10 = range_1,$ $range\text{-}size(lexeme) = 8 = range_2,$ $penalty_7 = 10 \times (6 - \frac{1}{2}((10-6)+(8-6)))/20 = 1.5$

So the overall proximity between the frame ($concept_i$) and the lexeme ($lexeme_j$) is calculated through $\sum_i penalty_i \times importance_i$ with calculated $penalty_i$ (conjunctive merging is used and if not defined $importance_i$ is taken as 1).

$$\begin{aligned} total\text{-}penalty(lexeme_j) = {} & (10 \times 0.6) + (8 \times 1) + (2.5 \times 1) + \\ & (4 \times 0.8) + (1.6 \times 0.2) + (3 \times 0.4) + (1.5 \times 0.6) \\ = {} & 21.72 \end{aligned}$$

After calculating the penalty of each lexeme candidate for a TMR frame, this phase sorts the candidates in increasing order based on those penalties. If there is still ambiguity in selection, more than one lexeme have the minimum calculated penalty or the differences between the first candidates are lower than some predefined threshold, this phase uses the information about the speech situation to resolve this ambiguity. This is because the speech situation has also an influence over selection of words by the speaker. So, stylistic information such as formality, color, force, etc. and pragmatic information such as speaker's attitude available in TMR is checked with the pragmatic definition of each lexeme in the lexicon. This utilization also improves the quality of word selection in generation.

4.1.3 Selection Algorithm

Whenever the current frame for being processed is an instantiation of a concept in the ontology, then the lexical selection module is called by the main module to get an open-class lexeme corresponding to that frame. The first task of this module is to find out all candidate lexemes from the target lexicon by getting those entries who are also instantiations of the concept used in the definition of that frame. After obtaining the candidate list, the context-dependent selection module is activated to remove lexemes whose contextual requirements are not satisfied by the TMR frame. The remaining candidate lexemes are sent to the context-independent selection module which chooses the most appropriate word for that TMR frame by using meaning comparison and speech situation properties. In

cases when only a single lexeme remained in the candidate list after context-dependent selection, that word is directly sent as the chosen lexeme for the target language and context-independent selection is skipped to avoid halting with no candidate.

There are cases in which the lexical selection module cannot succeed in choosing the final word because of insufficient knowledge in the lexicon or the TMR. In these cases, the selection module activates another submodule, which is called *augmentor*, to get help from the user or to inform the designer of the lexicon about its failure. Activation of the augmentor is the only case in which the main module interacts with the user. The followings are the cases in which the augmentor is activated:

- No lexeme is found in the lexicon which inherits the concept that the TMR frame is built on, which means that a new lexical item is needed in the lexicon which corresponds to that concept.
- All candidates are eliminated in context-dependent selection. Either a new word from the target language is needed whose contextual requirements do not conflict with the current TMR frame, or the contextual constraints of some candidate lexemes in the lexicon should be relaxed.
- Every candidate gets a penalty which is higher than a predefined maximum threshold in context-independent selection. This distance threshold is used to ensure that the lexeme selected is somehow close to the meaning resides in the TMR frame. Generally a new word should be added to the lexicon. Otherwise, the definitions of some candidates should be revised to release their specifications somehow.
- There are more than one candidate remained after the context-independent selection. This means that neither the proximity calculation nor the speech situation test can reduce the number of candidates to one. Either the definitions of some candidates should be constrained to represent more specific meaning expressions, or the meaning representation in TMR should be made richer in contents.

With the two new submodules, the constructor of the list of candidate lexemes and augmentor, the flow of the lexical selection module can be explained by the following algorithm:

Candidate-List := *construct-candidates-list*
if *cardinality*(*Candidate-List*) = 0
then *lexical-augmentor*(*Candidate-List*)
else *Dependent-List* := *context-dependent-selection*(*Candidate-List*)
if *cardinality*(*Dependent-List*) = 0
then *lexical-augmentor*(*Dependent-List*)
else **if** *cardinality*(*Dependent-List*) = 1
then *return*(*Dependent-List*[1])
else *Independent-List* := *context-independent-selection*(*Dependent-List*)
if *cardinality*(*Independent-List*) > 1
then *lexical-augmentor*(*Independent-List*)
else *return*(*Independent-List*[1])

The architecture of the lexical selection module together with its flow and utilization of the lexicon, the input TMR frame, and the overall TMR, is described in Figure 4.2.

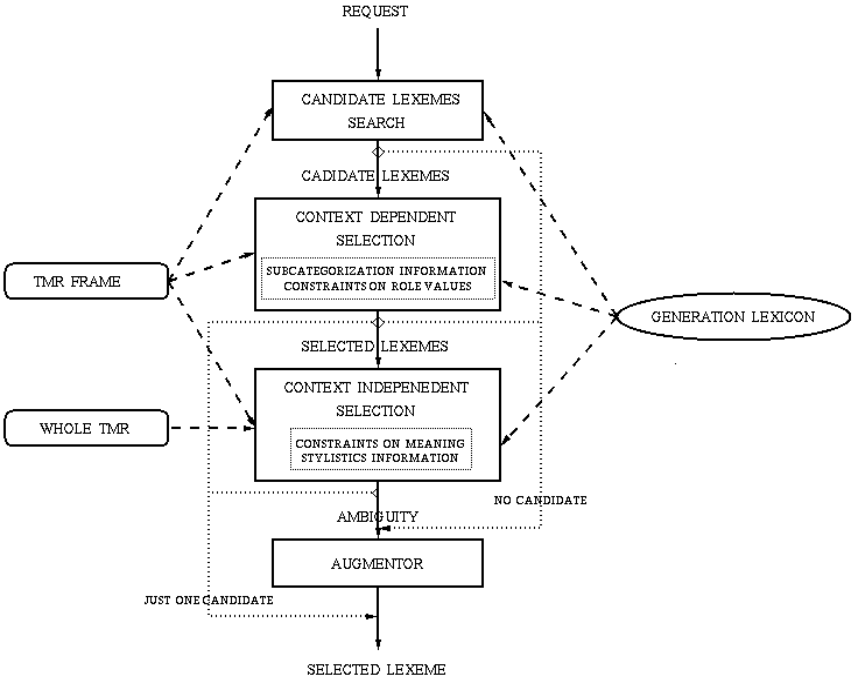

Figure 4.2: Lexical Selection Module

4.2 Map-Rules Application Module

The task of this module is to map the meaning representation in a TMR to a frame-based grammatical representation, feature structure, of the target language without lexical selection task. In order to achieve this task, the module makes use of the map-rules written for that language which are introduced to the

main module as a knowledge resource. As explained, map-rules are language specific knowledge about the relationship between the meaning patterns in TMR representation and the syntactic structure of the target language. Their applications, which depend on certain meaning contexts, change the syntactic knowledge created to be used by the tactical generation. So, this module is responsible for collecting all map-rules applicable to a TMR frame, processing them in a specific order, and updating the f-structure corresponding to the target sentence.

There are two major categories of TMR frames when map-rules are applied on them depending on whether the TMR frame is an instantiated concept from the ontology, or not. The second category includes all frames which are not instantiated concepts such as aspect, modality, focus, speech-act, etc. A TMR frame from the first category, an instantiated concept, is processed by map-rules written in two resources. The first resource is a set of map-rules written for the open-class lexical item selected for that frame. The map-rules created for the concepts in the ontology are the second source. All the map-rules written for a concept used in the frame's definition and the concepts that are ancestors of that concept are applied in this module. The ancestor concepts are used since the inheritance mechanism in the ontology is also utilized in the design of map-rules. The map-rules retrieved from these two sources are processed in a bottom-up fashion, from specifics to generals. In other words, the module starts with map-rules associated with the selected lexeme, and applies map-rules written for the concepts that are reachable from the concept used in instantiation by following *is-a* relations in the ontology. In traversing the ontology, the module makes the applications in breath-first order. A TMR frame from the second category is processed by map-rules written for just its frame type. That is, the module uses only the map-rules for speech-act when the frame to be processed is a speech-act frame. The following algorithm describes the overall behavior of the map-rule application module.

if *Current-Frame* is an *Instantiated-Concept*
then *Map-Rules-List* := *get-maps*(*Lexical-Item*)
  Current-FS := *apply-maps*(*Map-Rules-List*, *Current-FS*)
  Hierarchy-List := *parent*(*Lexical-Item*)
  while *Hierarchy-List* $\neq$ *NIL* **do**
    Map-Rules-List := *get-maps*(*first*(*Hierarchy-List*))
    Current-FS := *apply-maps*(*Map-Rules-List*, *Current-FS*)
    New-Hierarchy-List := *parents*(*first*(*Hierarchy-List*))
    Hierarchy-List := *append*(*New-Hierarchy-List*, *Hierarchy-List*)
else *Map-Rules-List* := *get-maps*(*Frame-Type*)
  Current-FS := *apply-maps*(*Map-Rules-List*, *Current-FS*)

In the algorithm above, *get-maps* collects the map-rules associated with its input argument from the map-rules knowledge resource. The input of *get-maps* can be a lexical item, a concept, or a frame used in TMR. The *apply-maps* routine processes the list of map-rules to update the current f-structure being constructed. Figure 4.3 shows the flow of the map-rule application module with its relation with the knowledge resources and the input TMR.

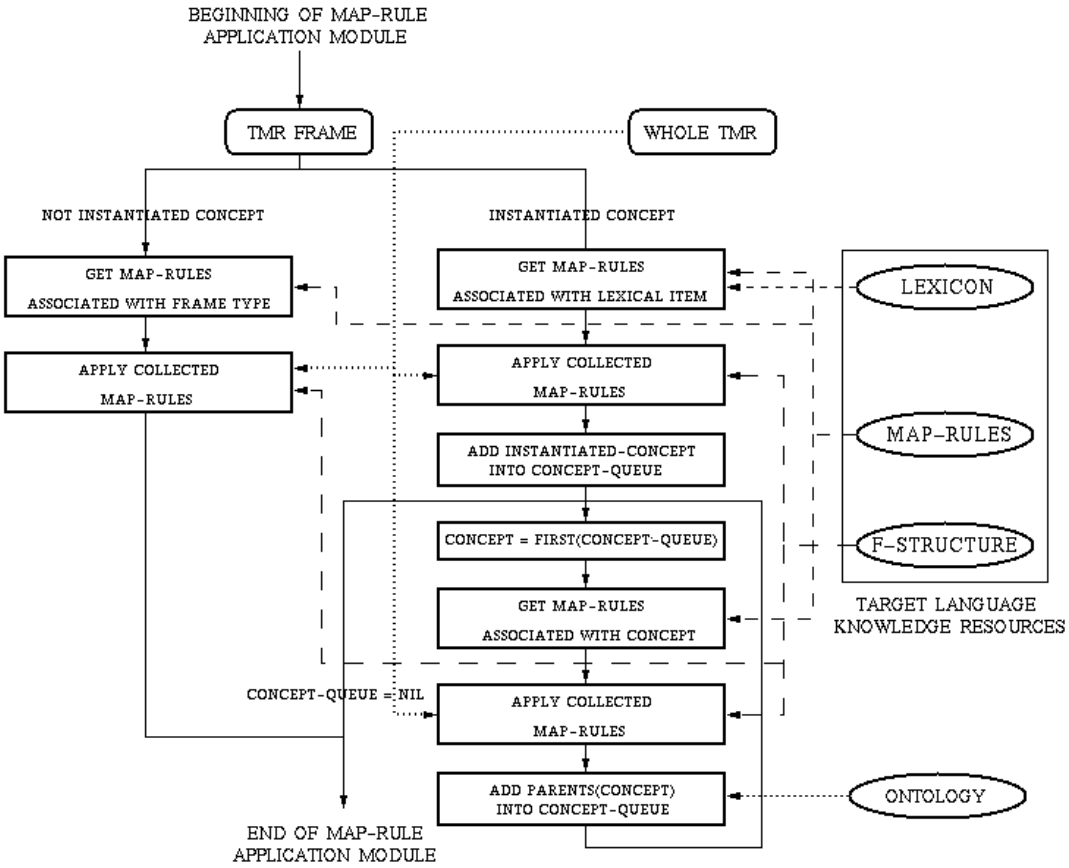

Figure 4.3: Map-Rule Application Module

Processing of a map-rule in *apply-maps* is done in two steps. First the meaning requirements for that map-rule are checked in the TMR. If the check is successful, then the f-structure is updated by the new information defined in that map-rule. Since the first step involves searching for specific knowledge in the TMR, and the second step is the way how f-structure is constructed and updated, these two phases are analyzed separately in the following two subsections.

4.2.1 Meaning Requirements Check

This phase checks the meaning requirements of a map-rule from the input TMR (TMR_{in}). As explained in Chapter 3, the context requirements of a map-rule are represented as a list of references to frames, their slots, and their features, that can exist in a TMR. To apply the f-structure update operations of a map-rule, every reference in the requirements list must be found in TMR_{in}. So, it is enough to check each reference iteratively to find the required meaning context of a map-rule. This phase processes the references individually in the given order, and it continues with the next one after finding a reference in TMR_{in}. Whenever a reference is not found in TMR_{in}, this phase halts with the failure of that map-rule, and the remaining references are not checked. Otherwise, it means that all references are available in TMR_{in} and the current map-rule's update operations can be applied.

There are two types of meaning context processing: *exclusive* and *any*. *Exclusive* type is introduced to group a set of dependent map-rules, and the application of one of them excludes the application of others. So, the meaning requirements of each map-rule grouped under *exclusive* type are checked individually until the meaning requirements of a map-rule from that set are satisfied by the input TMR. After finding the successful map-rule, the remaining map-rules in that set are not checked because of their exclusive property. Since map-rules grouped under *exclusive* type are checked in an iterative manner by the order defined in the *exclusive* set, the developer should guarantee that this order is appropriate for that grouping. *Any* type is introduced to group a set of independent map-rules. So, the meaning requirements of each map-rule from an *any* set are checked individually, and on success its f-structure update operations are performed without affecting the others in that set.

Each reference method described in Chapter 3 has a special processing module that is activated in this phase accordingly. For example, *ref(frame, feature,value)* calls a module which checks the existence of *(frame,feature,value)* in TMR_{in}, or *aspect(frame,[value$_1$,...,value$_n$])* activates a module which finds the *frame*'s aspect slot in TMR_{in}, assume that it is *aspect$_i$*, and checks the existences of *(aspect$_i$,Feature$_1$,value$_1$),...,(aspect$_i$,Feature$_n$, value$_n$)* iteratively in TMR_{in}. Note that *Feature$_1$,...,Feature$_n$* are variables unified with the feature names in *aspect$_i$*. One of the problems in this phase is to find the names of the frames in TMR_{in} which are referred through imaginary names like $Frame_{processing}$. This problem is solved through correspondences (such as $(Frame_{event} = BREAK_1)$) which are stored and dynamically updated by the main module.

4.2.2 Application of F-Structure Update Operations

After the meaning context required by a map-rule is satisfied by the input TMR, the second phase, which performs f-structure update operations of that map-rule, is activated. It processes each operation iteratively through using the f-structure representation of the target language introduced as a knowledge resource to the main module. It utilizes this resource to find the defined place of a *slot* or a $feature/value$ pair in the frame-based representation of f-structure since the place information is not provided in the definition of a map-rule (only the names of *slot*s or $feature/value$ pairs are given, Chapter 3). The three types of update operations (*frame-to-slot mapping*, *feature addition*, and *slot addition*) which are explained in Chapter 3 require different types of processing.There is a separate processing module for each of them.

Frame$_{TMR}$*-to-slot*$_{FS}$ *mapping* operations provide the connection between the semantic roles and the grammatical functions (such as mapping of *agent* to *subject*). These operations may not change the f-structure being constructed (FS), but gives information that can be utilized in future processing. There are two cases in processing this type of operations. The first case means that $frame_{TMR}$ is processed previously by the main module and a set of feature/value pairs and slots are inserted into a temporary f-structure ($frame_{FS}$) which cannot be connected to FS because of this missing map. These situations are handled through inserting $frame_{FS}$ into $slot_{FS}$ (connection achieved) and updating FS such that it has $slot_{FS}$. The second case is occurred when $frame_{TMR}$ is not processed at that time. These case is handled by creating a dynamic knowledge that informs the future processing about the map between $frame_{TMR}$ and $slot_{FS}$. This information will be used in the future while the main module processes $frame_{TMR}$ through other map-rules.

Feature addition is the most general operation utilized in f-structure creation and it is used to either introduce a new feature/value pair or update the value of a previously inserted feature. This operation is achieved by using the feature structure representation of the target language provided as a knowledge resource to the main module. The frame-based notation used for the explanation of f-structure in Chapter 3 is represented as a multi-parent tree in this knowledge resource. In this representation, slots are denoted as the internal nodes of the f-structure tree and feature/value pairs are represented as the leaves of that tree. Figure 4.4 describes this proposed representation.

Features and slots in an f-structure representation describe distinct syntactic

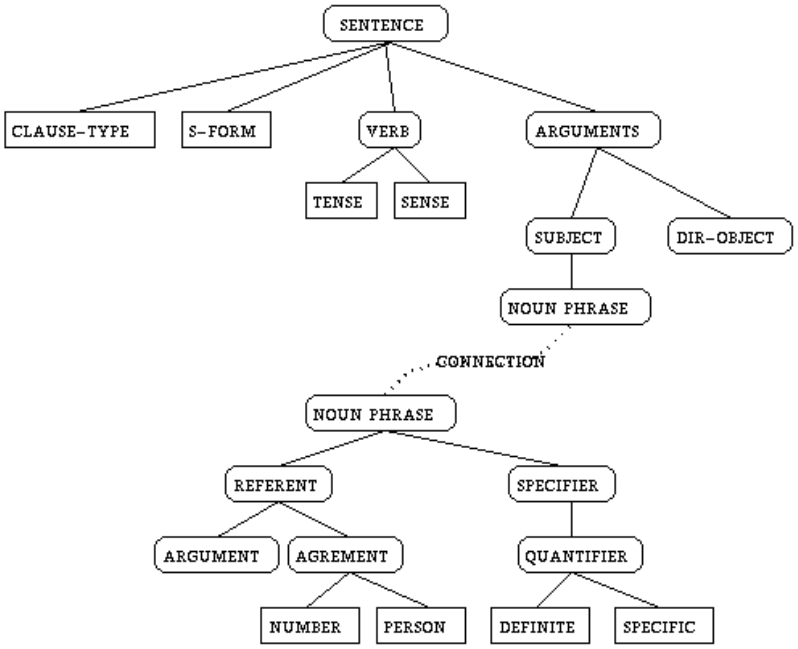

Figure 4.4: F-Structure Representation

phenomenon in a natural language. So, they should be named uniquely, otherwise they can be confused. Since the leaf node that contains a feature/value pair should be found to insert that pair into an f-structure, a heavy top-to-bottom search is needed in both the frame-based and the tree representation of f-structure. But the uniqueness property mentioned above can be utilized to overcome this inefficiency. In this work, the proposed tree representation is improved such that the place of a slot or a feature/value pair are found directly (indexing). After finding the place, only a bottom-up traversal in the tree is required to update the f-structure.

Although the proposed representation is efficient, there is still one problem that is not solved. Remember that, the arguments of a verbal phrase such as *subject*, *direct-object* and *time* are noun phrases in Turkish. So, the utilized uniqueness property does not hold for all cases. To handle this, slots that can be used in the definition of more than one syntactic construction are represented as individual trees in this work. In other words, there are two trees in the representation of Turkish sentences: one for the verbal phrases and one for the noun phrases. In this way, the uniqueness property is recovered.

Although there are more than one tree in the proposed representation, one of them ($MAIN$) is not a child of others (after all, the overall structure is also a tree). Since the child trees should be connected to $MAIN$ in the final constructed f-structure, a new information is needed (there are more than one place that child

tree can be inserted). This information is provided by *frame-to-slot mapping* rules which bound the results of some frames to specific places in the f-structure. So, there are three cases that should be handled differently in an feature addition operation:

1. The feature/value pair to be inserted is in $MAIN$. Since there is no ambiguity about the place, that feature/value is found as a leaf in $MAIN$. Then, $MAIN$ is traversed in a bottom-up fashion starting from the node that contains the feature/value pair. Traverse is ended when the root of $MAIN$ is reached. For example, in Figure 4.4, to insert $sense/positive$, firstly it is found as a child of node $verb$, which is a child of root. So, the traversal ends with the structure $[verb, [[sense, positive]]]$.

2. The feature/value pair to be inserted is in a child tree. That feature/value pair is found as a leaf of that tree, but the traversal ends with a root different from $MAIN$. If there is a map information produced in the previous flow of the processing, then this knowledge is used to make the connection between the child tree and $MAIN$. For example, if $person/third$ is to be inserted into the tree in Figure 4.4, then first $[referent, [[agr, [[person, third]]]]]$ is constructed. Finding that current root is not $MAIN$, (assuming that $frame_{TMR}$ is being processed currently) it is checked whether there is a map information about $frame_{TMR}$. Finding that it is mapped to $subject$ in $MAIN$, previously constructed structure is inserted into $subject$ and traversal continues in $MAIN$. Finally, the following structure is produced from this update request: $[arguments, [[subject, [[referent, [[agr, [[person, third]]]]]]]]]$.

3. The feature/value pair to be inserted is in a child tree and there is no map information. Since, the structure that is constructed in the child tree should not be wasted, this type of cases are handled by creating a dynamic knowledge that the constructed structure is produced from the process of the current TMR frame. This information can be used in future if the missing map information is produced by other map-rules. So, for the example given above, an information is created such that $[referent, [[agr, [[person, third]]]]]$) is constructed from the processing of $frame_{TMR}$.

Upto this point, the insertion of a feature/value pair into an empty f-structure is explained. In general, insertions are done into partially created

f-structures. In this work, instead of changing the contents of the f-structure being constructed directly, a merge operation is utilized to improve efficiency (to avoid top-to-bottom search in the trees) during insertion. This merge operation is activated whenever an f-structure operation creates a new substructure that is to be inserted into the main f-structure. It also uses the uniqueness property of an f-structure representation. There are four cases that should be handled in this merge operation:

1. The newly created structure contains only a feature/value pair and it is not found in the main f-structure. In this case, the new structure is directly appended to the main one.

2. The newly created structure contains only a feature/value pair and that feature is inserted previously to the main f-structure. In this case, the value of that feature is updated with its new value.

3. The parent slot in the newly created structure is found in the main f-structure. Merge operation continues with the contents of that slots recursively as if they are the structures to be merged. After this insertion is achieved, the content of the main f-structure (except that slot) is appended to the result of that insertion.

4. The parent slot in the newly created structure is not found in the main f-structure. In this case, the content of the new structure is directly appended to the main f-structure.

Slot addition can be performed by the techniques described for feature addition and it is handled in the same way a feature addition operation is performed.

4.3 Main Module

The main module of the computational module can be separated into two independent consecutive operations. Processing the frames in the input TMR is done in depth-first manner, which guarantees that a frame with all its children are processed before any other frame in the TMR. Depth-first processing is utilized in processing TMRs that have more than one event in their contents and explained at the end of this section. The first step constructs the initial processing stack which is filled with the following frames in the given order:

- *Main Event*: The event that is the scope of the speech-act.
- *Top Relations*: Relations that connect the decomposed meanings in the frames such as temporal relations, domain relations, coreference information, etc. (obtained from *table-of-contents*)
- *Events*: List of all events in the TMR that are used in the definition of the overall propositional meaning (excludes *main event*). The *events* list is also obtained from *table-of-contents*.

Top relations are put before the *events* since they can relate available events to the *main event*. The stack is updated such that a frame can not be inserted more than once, and the most current one determines the processing place, the old ones are deleted. Sometimes, a frame is processed not directly, but during the process of another frame. Also, in this case, that frame should be deleted from the stack. These two requirements are handled through maintaining a processed frames knowledge, which is a list of frames that are processed until the current processing stage. By using this list, a processed frame in the stack is directly deleted without reprocessing.

After the initialization step is completed, the main phase parses the overall meaning representation of a sentence. This phase continues until the processing frame stack becomes empty. Each frame in the stack is processed through the application of the lexical selection and the map-rules application modules, then it is removed from the stack and added into the processed frames list. Then, the frames that are children of this frame are inserted into the processing stack and parsing continues on the next frame. The following algorithm describes the overall behavior of the main module:

F-Structure := *NIL*
Processed-List := *NIL*
Processing-Stack := *create-processing-stack*
while *Processing-Stack* $\neq$ *NIL* **do**
    Processing-Frame := *pop*(*Processing-Stack*)
    if *Processing-Frame* is not in *Processed-List*
    then *F-Structure* := *process-frame*(*Processing-Frame*, *F-Structure*)
        New-Processing-Frames := *get-child-frames*(*Processing-Frame*)
        Processing-Stack := *push*(*Processing-Stack*, *New-Processing-Frames*)
        Processed-List := *insert*(*Processing-Frame*, *Processed-List*)
return *F-Structure*

The processing of instantiated concept frames is different from the processing of other frames. If a frame is an instantiated concept, then the lexical selection module is activated which returns the chosen open-class word for that concept. Then, f-structure features which are defined for that lexeme in the lexicon are directly inserted into the constructed f-structure. Finally, the map-rules application module is called to update the constructed f-structure for that frame. Frames of the second type are directly sent to the map-rule application module. The following algorithm shows how the process of a single frame proceeds in the parse phase.

if *Processing-Frame* is a $CONCEPT$
then $Lexical\text{-}Item := lexical\text{-}selection\text{-}module(Processing\text{-}Frame)$
$F\text{-}Structure := insert\text{-}lexical\text{-}features(Lexical\text{-}Item, F\text{-}Structure)$
$F\text{-}Structure := map\text{-}rules\text{-}application\text{-}module(Processing\text{-}Frame, F\text{-}Structure)$
else $F\text{-}Structure := map\text{-}rules\text{-}application\text{-}module(Processing\text{-}Frame, F\text{-}Structure)$

The architecture of the main module is shown in Figure 4.5 in which the flow, usages of the submodules, their relationships with the knowledge sources, and their effects on the processing information are described.

One thing that is not explained until here is how the events available in a TMR are connected in the f-structure, that is how the main module processes a TMR if there is more than one event in it. The need for special treatment arises from the fact that every event in a TMR results in an individual f-structure representation (structures that are rooted in $MAIN$, see Section 4.2.2). Since there are more than one f-structure, the main module should decide on the f-structure an update operation is performed. Depth-first processing is utilized here which guarantees that every event with all its child frames in a TMR processed individually. The main module should also constructs the final f-structure through merging those individual f-structures as sentential clauses or constructing a complex-sentence described in Chapter 3. There are three different ways in which the events in a TMR are related:

- *Thematic Role*: Covers the cases in which an event fills the thematic role of another event. "I want to read the books of Faucault." is an example for this type of relation in which there are two events, $WANT$ and $READ$, and $READ$ fills the thematic role *theme* of $WANT$.

- *Domain Relation*: Covers the cases in which two events are related through a domain relation in a TMR. For example, in sentence "Since Ali didn't study enough, he couldn't pass the exam.", there is a causal relationship

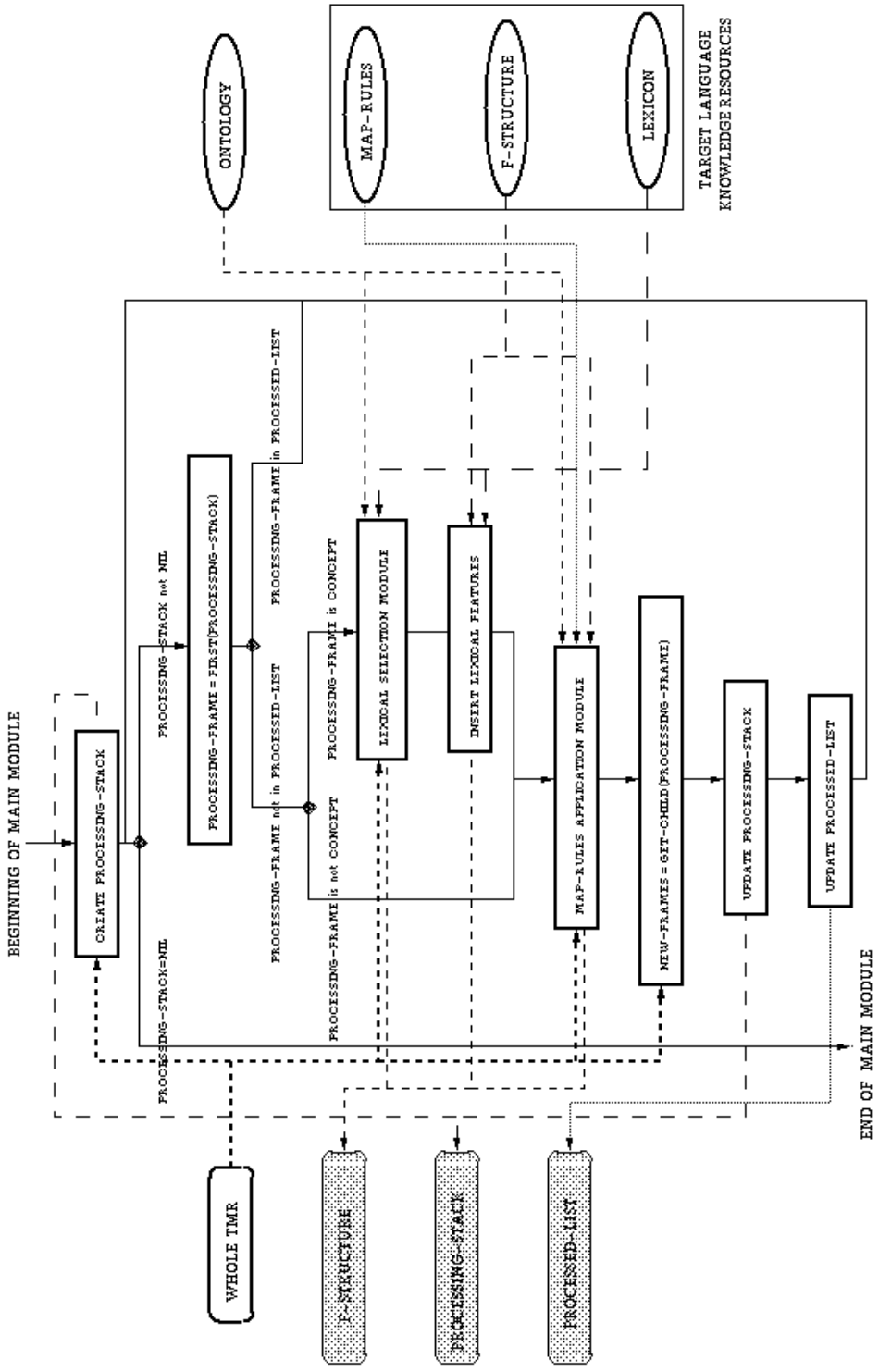

Figure 4.5: Main Module of Computational Model

between the two events $STUDY$ and $PASS$ and it is represented in the TMR as $domain\text{-}rel(reason, STUDY, PASS)$.

- *Contextual Boundedness*: Covers the cases in which one event is introduced to give some contextual information about another event or its components. In these cases, these two events should have some common property such as each event is performed by the same agent, they occur at the same time, or in the same location, etc. The sentence "Ali, who came to your birthday party last month, went to America." is an example for contextual boundedness. The two events $COME$ and GO are related to each other through the same $agent$ and the event $COME$ is used as a definite description of the $agent$ of the main event GO.

The methodology utilized in this work is to process the main event of the expression represented in the input TMR before any other events and determine the relationship between the events afterwards. Each event other then the main event causes the main module to restart with an empty f-structure and processing stack filled with the new event. After constructing the f-structure corresponding to that event, halting with empty processing stack, that f-structure is connected with the f-structure created for the main event. So, handling each relation type is achieved as follows:

- *Thematic Role*: The f-structure created for the event which fills the thematic role of the main event is inserted into the main f-structure through using slot-to-slot mapping rules. In the sentence "I want to read the books of Faucault.", assuming that $theme$ is mapped to $dir\text{-}object$, FS_{READ} is inserted into the $dir\text{-}object$ argument of the event $WANT$ as a sentential clause ($merge\text{-}events$ in the algorithm below).

- *Domain Relation*: A complex sentence is constructed from the FS_{main} and the FS_{event} through using the map-rules written for the type of the domain relation. So, in the sentence "Since Ali didn't study enough, he couldn't pass the exam.", FS_{STUDY} is connected to FS_{PASS} by $[[type, linked], [relation, 'için'], [arg_1, FS_{STUDY}], [arg_2, FS_{PASS}]]$ ($activate\text{-}contextual\text{-}maps$ in the algorithm below).

- *Contextual Boundedness*: The f-structure created for FS_{event} is placed into the $slot_{FS}$ of the FS_{main} which is created for the description of the common context with the content of $slot_{FS}$ is moved into FS_{event}. So, in the sentence "Ali, who came to your birthday party last month, went to

America" the content of *subject* in the FS_{GO} is moved into the *subject* of the FS_{COME} and this new structure is inserted into the *subject* of FS_{GO} (*activate-domain-type-maps* in the algorithm below).

The following algorithm is used to handle events other than the main one through the methods explained for each type of relation:

if *Processing-Frame* is an *event*
then *Event-Processing-Stack* := $list(Processing\text{-}Frame)$
Event-F-Structure := $main\text{-}module(Event\text{-}Processing\text{-}Stack)$
if *Processing-Event* is bound to a $slot_{FS}$ in *F-Structure*
then *F-Structure* := $merge\text{-}events(Event\text{-}F\text{-}Structure, slot_{FS}, F\text{-}Structure)$
else *F-Structure* := $activate\text{-}contextual\text{-}maps(Event\text{-}F\text{-}Structure, F\text{-}Structure)$
else if *Processing-Frame* is a $domain\text{-}rel(type, Event_{main}, Event_{other})$
then *Event-Processing-Stack* := $list(Event_{other})$
Event-F-Structure := $main\text{-}module(Event\text{-}Processing\text{-}Stack)$
F-Structure := $activate\text{-}domain\text{-}type\text{-}maps(Event\text{-}F\text{-}Structure, F\text{-}Structure)$

4.4 An Example

To get a clear understanding of how the computational model described in this chapter processes an input TMR, an easy example from Turkish is given. The sentence chosen for this example is "Kadın camı kırdı." which is translated into English as "The woman broke the window.". The activation of the submodules, the lexical selection module and map-rules application module, by the main module, and their effects on the f-structure are given through the example with sample map-rules and lexicon entries. The TMR representation of this sentence is the following:

table-of-contents

speech-act	$speech\text{-}act_1$
heads	$BREAK_1$
time-rels	$temp\text{-}rel_1$
attitudes	NIL
modalities	NIL
focus	NIL
stylistics	NIL
coreferences	NIL
domain-rels	NIL

$BREAK_1$	
agent	$HUMAN_1$
patient	$WINDOW_1$
polarity	*positive*
aspect	$aspect_1$
time	$time_1$
$aspect_1$	
phase	*perfect*
iteration	*single*
duration	*momentary*
telicity	*false*
$speech\text{-}act_1$	
type	*declarative*
scope	$BREAK_1$
speaker	*speaker*
hearer	*hearer*
time	$time_2$

$HUMAN_1$	
type	*common*
gender	*female*
age	≥ 17
reference	*definite*
$WINDOW_1$	
reference	*definite*
$time_1$	
absolute	*past*
$time_2$	
absolute	*past*
$temp\text{-}rel_1$	
type	*after*
arg_1	$time_2$
arg_2	$time_1$

In this example, the main event is represented with $BREAK_1$ whose *agent* is defined as $HUMAN_1$ and *patient* as $WINDOW_1$. The aspectual properties of the event $BREAK_1$ is represented in $aspect_1$, whose content is determined by the information that 'kır' is a punctual event. The event $BREAK_1$ was occurred before the time of speech and this information represented in $temp\text{-}rel_1$. Note that, since the expression refers to known entities which fills the *agent* and the *patient*, both $HUMAN_1$ and $WINDOW_1$ have the feature $(reference, definite)$. The main module starts with the following initializations:

$$\begin{aligned} Processing\text{-}Stack &= [BREAK_1, temp\text{-}rel_1, speech\text{-}act_1] \\ Processed\text{-}Frames &= [\,] \\ F\text{-}Structure &= [\,] \end{aligned}$$

The first frame in *Processing-Stack*, $BREAK_1$, is extracted as *Processing-Frame*. Since $BREAK_1$ is an instantiation of a concept, first the lexical selection module is activated. From Turkish lexicon, the lexical selection module chooses the entry $kır_1$ which is a child of the concept $BREAK$ with the following definition:

$kır_1$		
CAT	*category*	*verb*
	root	*kır*
SEM	*is-a*	*BREAK*
	subcat-info	
	requires	[*patient*]
	optional	[*agent*, *means*]

Since it is the only candidate, it is directly sent to the main module as the selected lexeme (note that subcategorization constraints are satisfied by $BREAK_1$). Since *Processing-Frame* is an *event*, the default features $[category, verb]$ and $[root, kır]$ are inserted into the *verb* slot and *F-Structure* becomes:

$$F\text{-}Structure \quad = \quad [verb, [[category, verb], [root, kır]]]$$

Then, map-rules written for $kır_1$ are collected from the Turkish map-rules and only the following rule succeeds:

$$\begin{aligned} &maprule(turkish, kır_1, exclusive, \\ &\quad [exist(Frame_{processing}, agent, Slot_1), \\ &\quad\ exist(Frame_{processing}, patient, Slot_2), \\ &\quad\ not\text{-}exist(Frame_{speech}, focus)] \\ &\quad [map(Slot_1, subject), map(Slot_2, dir\text{-}object)]) \end{aligned}$$

$Slot_1$ is unified with $HUMAN_1$ and $Slot_2$ is unified with $WINDOW_1$ and the following match information is produced by the map-rules application module.

$$\begin{aligned} &map(HUMAN_1, subject) \\ &map(WINDOW_1, dir\text{-}object) \end{aligned}$$

The map-rule application module starts traversing the ontology for map-rules associated with concepts from the concept *BREAK*. Map-rules for *BREAK* fail, and the following map-rule written for *PUNCTUALITY*, the only parent of *BREAK*, succeeds and it updates both the *F-Structure* and the *Processed-Frames* (the whole contents of $aspect_1$ and $temp\text{-}rel_1$ are checked, so they are declared to be processed).

$$\begin{aligned} &maprule(turkish, PUNCTUALITY, exclusive, \\ &\quad [aspect(Frame_{processing}, [perfect, single, momentary, false]), \\ &\quad\ time(after, [Frame_{speech}, Frame_{processing}])], \\ &\quad [feature(tense, past)]) \end{aligned}$$

$$\begin{aligned} Processed\text{-}Frames &= [aspect_1, temp\text{-}rel_1] \\ F\text{-}Structure &= [verb, [[category, verb], [root, kır], [tense, past]]] \end{aligned}$$

The parent of *PUNCTUALITY* is *EVENT* and it has two map-rules that succeed for the input TMR with the following changes:

$maprule(turkish, EVENT, any,$
$\quad [ref(Frame_{speech}, scope, Frame_{processing})],$
$\quad [feature(clause\text{-}type, predicative)])$
$maprule(turkish, EVENT, any,$
$\quad [ref(Frame_{processing}, polarity, positive)],$
$\quad [feature(sense, positive)])$

$F\text{-}Structure =$
$\quad [[clause\text{-}type, predicative],$
$\quad [verb, [[category, verb], [root, kır], [tense, past], [sense, positive]]]]$

The following map-rule written for ALL, the parent of $EVENT$, succeeds and updates the $F\text{-}Structure$.

$maprule(turkish, ALL, any,$
$\quad [same(Frame_{processing}, Frame_{event})],$
$\quad [feature(s\text{-}form, finite)])$

$F\text{-}Structure =$
$\quad [[clause\text{-}type, predicative], [s\text{-}form, finite],$
$\quad [verb, [[category, verb], [root, kır], [tense, past], [sense, positive]]]]$

Since ALL is reached, the map-rule application module halts, and the child frames of $BREAK_1$, which are $HUMAN_1$, $WINDOW_1$, $aspect_1$, and $time_1$, are inserted to $Processing\text{-}Stack$. Note that, since $aspect_1$ is processed in the process of $BREAK_1$, it is not inserted, and the state of the main module becomes:

$Processing\text{-}Stack = [HUMAN_1, WINDOW_1, time_1, temp\text{-}rel_1, speech\text{-}act_1]$
$Processed\text{-}Frames = [BREAK_1, aspect_1, temp\text{-}rel_1]$

The next frame to be processed is $HUMAN_1$ and the lexical selection module is activated which finds four entries from the Turkish lexicon that are instantiations of $HUMAN$.

$adam_1$			$kadın_1$		
CAT	$category$	$noun$	CAT	$category$	$noun$
	$root$	$adam$		$root$	$kadın$
SEM	$is\text{-}a$	$HUMAN$	SEM	$is\text{-}a$	$HUMAN$
	$definition$			$definition$	
	$type$	$common$		$type$	$common$
	$gender$	$male$		$gender$	$female$
	age	≥ 17		age	≥ 17

$çocuk_1$			$bebek_1$		
CAT	$category$	$noun$	CAT	$category$	$noun$
	$root$	$çocuk$		$root$	$bebek$
SEM	$is\text{-}a$	$HUMAN$	SEM	$is\text{-}a$	$HUMAN$
	$definition$			$definition$	
	$type$	$common$		$type$	$common$
	$gender$	$unknown$		$gender$	$unknown$
	age	≥ 4 & ≤ 11		age	≤ 4

Since $kadın_1$ gets no penalty because of exact match, it is returned as the selected lexeme. Its default features, $[category, noun]$ and $[root, kadın]$, are inserted into the argument $subject$ because of the knowledge $map(HUMAN_1, subject)$.

$$
\begin{array}{l}
F\text{-}Structure = \\
\quad [[clause\text{-}type, predicative], [s\text{-}form, finite], \\
\quad\ [verb, [[category, verb], [root, kır], [tense, past], [sense, positive]]], \\
\quad\ [arguments, [[subject, [[referent, [[arg, [category, noun], [root, kadın]]]]]]]]
\end{array}
$$

No map-rule associated with lexeme $kadın_1$ succeeds, and map-rule application module starts traversing the ontology from the concept $HUMAN$. Following map-rule written for $HUMAN$ succeeds and updates the $F\text{-}Structure$.

$$
\begin{array}{l}
maprule(turkish, HUMAN, exclusive, \\
\qquad [ref(Frame_{processing}, type, common)], \\
\qquad [feature(number, singular), feature(person, third)])
\end{array}
$$

$$
\begin{array}{l}
F\text{-}Structure = \\
\quad [[clause\text{-}type, predicative], [s\text{-}form, finite], \\
\quad\ [verb, [[category, verb], [root, kır], [tense, past], [sense, positive]]], \\
\quad\ [arguments, [[subject, [[referent, [[arg, [[category, noun], [root, kadın]]], \\
\qquad\qquad\qquad\qquad [agr, [[number, singular], [person, third]]]]]]]]]
\end{array}
$$

The following map-rule associated with $ENTITY$, the parent of $HUMAN$ in the ontology, is successfully applied and the $F\text{-}Structure$ becomes:

$$
\begin{array}{l}
maprule(turkish, ENTITY, any, \\
\qquad [ref(Frame_{processing}, reference, definite)], \\
\qquad [feature(definite, positive)])
\end{array}
$$

$$
\begin{array}{l}
F\text{-}Structure = \\
\quad [[clause\text{-}type, predicative], [s\text{-}form, finite], \\
\quad\ [verb, [[category, verb], [root, kır], [tense, past], [sense, positive]]], \\
\quad\ [arguments, [[subject, [[referent, [[arg, [[category, noun], [root, kadın]]], \\
\qquad\qquad\qquad\qquad [agr, [[number, singular], [person, third]]]]], \\
\qquad\qquad\qquad [specifier, [[quan, [[definite, positive]]]]]]]]]
\end{array}
$$

Map-rules written for ALL are not applicable to $HUMAN_1$, which ends the processing of the map-rule application module. Since $HUMAN_1$ has no child frame, it is added to *Processed-Frames* and *Processing-Stack* remains unchanged.

$$Processed\text{-}Frames \;=\; [HUMAN_1, BREAK_1, aspect_1, temp\text{-}rel_1]$$

The next frame from *Processing-Stack* is $WINDOW_1$ and following operations are performed for it in the lexical selection and map-rules application modules.

SelectedLexeme :

cam_1			
	CAT	*category*	*noun*
		root	*cam*
	SEM	*is-a*	$WINDOW$

Map Information : $map(WINDOW_1, dir\text{-}object)$
Default Features : $[[category, noun], [root, cam]]$

Applied Map-Rules :

$maprule(turkish, ARTIFACT, any,$ $(ARTIFACT = parent(WINDOW))$
$[],$
$[feature(number, singular), feature(person, third)])$
$maprule(turkish, ENTITY, any,$ $(ENTITY = parent(ARTIFACT))$
$[ref(Frame_{processing}, reference, definite)],$
$[feature(definite, positive)])$

F-Structure =
$[[clause\text{-}type, predicative], [s\text{-}form, finite],$
$[verb, [[category, verb], [root, kır], [tense, past], [sense, positive]]],$
$[arguments, [[subject, [[referent, [[arg, [[category, noun], [root, kadın]]],$
$[agr, [[number, singular], [person, third]]]]],$
$[specifier, [[quan, [[definite, positive]]]]]]],$
$[dir\text{-}object, [[referent, [[arg, [[category, noun], [root, cam]]],$
$[agr, [[number, singular], [person, third]]]]],$
$[specifier, [[quan, [[definite, positive]]]]]]]]]]$

Since $WINDOW_1$ has no child frame, it is added to *Processed-Frames* and the next frame from *Processing-Stack*, $time_1$, is declared to be $Frame_{processing}$. It has neither applicable map-rules nor child frames. So, the next frame, $temp\text{-}rel_1$ is extracted which is skipped since it is in *Processed-Frames*. The next frame, *speech act*$_1$ causes application of the following map-rules.

$Applied\ Map\text{-}Rules$:

$maprule(turkish, speech\text{-}act, any,$
$[ref(Frame_{processing}, type, declarative)],$
$[feature(speech\text{-}act, declarative)])$
$maprule(turkish, speech\text{-}act, exclusive,$
$[not\text{-}exist(Frame_{processing}, focus)],$
$[feature(voice, active)])$

$F\text{-}Structure =$
$[[clause\text{-}type, predicative], [s\text{-}form, finite],$
$[speech\text{-}act, declarative], [voice, active],$
$[verb, [[category, verb], [root, kır], [tense, past], [sense, positive]]],$
$[arguments, [[subject, [[referent, [[arg, [[category, noun], [root, kadın]]],$
$[agr, [[number, singular], [person, third]]]]],$
$[specifier, [[quan, [[definite, positive]]]]]],$
$[dir\text{-}object, [[referent, [[arg, [[category, noun], [root, cam]]],$
$[agr, [[number, singular], [person, third]]]]],$
$[specifier, [[quan, [definite, positive]]]]]]]]$

Frame $time_2$, which is a child of $speech\text{-}act1$ is added to $Processing\text{-}Stack$. Like $time_1$, map-rules associated with $time$ are not applicable for the input TMR, and finally $Processing\text{-}Stack$ becomes empty which causes the main module to halt with the last shown $F\text{-}Structure$ as the target sentence. Note that, map-rules written for the concepts in the ontology provides abstraction in their construction and also avoids enumeration (map-rule written for $ENTITY$).

Chapter 5

Implementation

The computational model described in Chapter 4 is implemented in *Prolog*. There are two main reasons for choosing Prolog as the implementation language. The first reason is the *symbolic manipulation* requirement of the computational model. Symbolic manipulation is needed firstly in the map-rule application module to check the content of a TMR. It is also needed in the lexical selection module to compare the meaning of an instantiated concept frame in a TMR with the definitions of lexemes provided in target lexicon. So, symbolic manipulation is at the core of the computational model. The second reason is the *ease in knowledge-base construction and efficiency in retrieval* requirement. It is needed since the computational model is heavily based on knowledge resources (ontology, lexicon, map-rules, and f-structure syntax), which are utilized in processing the input TMR. Since Prolog is one of the programming languages which is powerful in both symbolic manipulation and knowledge-base construction, it is chosen as the implementation language.

The implemented Prolog program takes another knowledge resource as input which contains information about the languages that are currently available to the system and their lexicon and map-rules as knowledge resources. It takes the input TMRs from a file. There is a loading facility that is provided to load the knowledge resources of another language or another file that contains different TMRs. This loading facility first extracts all knowledge about the old language or the old TMR file to lower the memory requirements of the system. Also, a trace facility is provided which creates a report about how lexical selections are done and which map-rules are applied while processing a TMR.

This chapter describes the real format of an input TMR that is processed by the implemented Prolog program and explains how it is created in Section 5.1, defines how the knowledge resources are represented in Prolog in Section 5.2, and

finally gives the time complexity of the overall system in Section 5.3.

5.1 TMR Parser

Text meaning representation of a sentence, which is the input to the developed system, is created manually since currently there is no available tool that creates such an input. Although the representation described for TMR in Chapter 3 is very user friendly, ease in understanding and creation, it is very difficult to manipulate a TMR in that format. So, some improvements are made to the representation of TMR that are not covered previously. First, to improve efficiency in differentiating concepts from other values in a TMR, instantiated concept frames are represented with a % preceding the concept name, and direct concept references (see Chapter 3, in set frames) are represented with a * preceding the concept name. Second, to avoid losing such a user friendly representation, another tool, *TMR parser*, is developed to produce an equivalent but different representation, that can be efficiently used in Prolog, from TMR representation presented in Chapter 3. Third, to adopt the developed system to the output of a *text planner*, multiple input TMR processing capability is provided.

The implemented tool, *TMR parser*, that produces the real input TMRs from TMRs written in the format described in Chapter 3 achieves two major goals. The first goal is to eliminate the major disadvantage of manual construction, high probability in making mistakes while writing. The second goal is to produce an equivalent representation which is richer in content to improve efficiency and handle multiple input TMRs. So, the tool is divided into two phases: parsing textual input and producing the utilized Prolog representation.

The first phase takes the manually created textual input TMRs and produces an intermediate representation which can be utilized effectively by the second phase. This phase also produces a report about the syntactic mistakes found in the input TMRs. Parsing the textual input is achieved through using the definitions of concepts in the ontology and the definitions of linguistic (speech-act, modality, etc.) and special (set, table-of-contents, etc.) frames. These definitions are given to the parser as the syntax knowledge of TMR.

The description of a concept is extracted from the ontology by combining its thematic roles with its definition. Since features from parent concepts can be used in the instantiation of a concept, the complete definition of a concept

is obtained by getting the definitions of parent concepts and merging them with the concept's definitions. Remember that, the definitions of other frame types are based on some required feature/value pairs and this distinction is provided in the given syntax of TMR. Each frame type is described with a set of required feature/value pairs and some other optionals. For example, a *speech-act* frame must contain *type*, *scope*, and *time* features and can take *producer*, *consumer*, *modality*, and *focus* features. Note that, an event frame must contain *aspect*, *time*, and *polarity* features to provide its temporal properties and its truthness. An event frame can also take *modality* and *attitude* frames as optionals. So, this additional information is appended to the definition of a frame if it is used as the description of an event. Also, an entity frame can take *attitude* optionally and this information is embedded into the definition of a frame that denotes an entity.

Each TMR in the textual input starts with *table-of-contents* frame. This frame is used both in obtaining the general information (the list of event frames, whether there are domain relations or stylistic information) about that TMR. It is also used to determine the beginning of the next TMR. A frame list, that contains frame names that are referenced in processing but not defined yet in the TMR, is constructed initially from *table-of-contents* and updated every time a new frame is parsed. A frame that is completely parsed is extracted from this list. Each frame in the TMR is processed by getting its definition, reading its content from the text and comparing the feature/value pairs which reside in input TMR with its definition. Currently, seven sources of TMR mistakes are handled in this phase:

1. *table-of-contents* frame is not found, which is not allowed. So, it is handled by skipping all frames until a *table-of-contents* frame is found and reporting a fatal error that those frames are skipped.
2. The name of an input frame is not found among concepts and TMR frame types. Since there is no available definition for that frame name, only its feature/value pairs with its name are converted into symbolic values and no check about their validity is done. This error is reported such that the name of that TMR frame is invalid.
3. A required feature for a frame type is not found. An error message is produced that the feature is required for that frame type.
4. A feature in the input TMR frame is not found in the definition provided. In this case, that feature with its value are converted into symbolic values

and appended to the intermediate representation with a prompt that the feature is not found in the definition.

5. The given value of a feature is not available in the feature domain. The method utilized in the third item is applied with an error message that the input value is out of the domain.
6. A frame is not defined in the whole TMR although it is referenced in some other frame (at the end of a TMR, there is still frames in the processed list). An error message is produced such that the frame is not found in input TMR.
7. A defined frame is not referenced in the other frames of a TMR. In this case, an error message is produced such that the defined frame is not referenced in the whole TMR.

Each frame is represented as a list of feature/value pairs, including its frame type, its frame index, and its content. Since there can be multiple TMRs, each TMR in the textual input is separated from others by representing it as an individual list. So, the output of this phase is a set of lists that represents the TMRs in the textual input. The output of this phase for an individual TMR is the following list structure which can be easily manipulated in Prolog.

$$\left[\begin{array}{l} \left[\begin{array}{ll} type & FrameType_1 \\ id & FrameIndex_1 \\ feature_1 & value_1 \\ \vdots & \vdots \\ feature_m & value_m \end{array}\right]_{Frame_1} \\ \\ \vdots \\ \\ \left[\begin{array}{ll} type & FrameType_n \\ id & FrameIndex_n \\ feature_1 & value_1 \\ \vdots & \vdots \\ feature_m & value_m \end{array}\right]_{Frame_n} \end{array}\right]_{TMR_i}$$

Although this intermediate representation can be used as the input TMR to the developed system, it is still somehow inefficient if the retrieval of a frame with its feature/value pairs is considered. This is because a search is needed to find the needed frame in the TMR list, and even a search is required to get a feature/value pair in a frame of the input TMR. Also, determining whether an

instantiated concept frame denotes an event or an entity in the input TMR is a time consuming job, since *table-of-contents* frame must be found to obtain the list of event frames. Finding the children frames of a frame, which is required in updating the processing stack, is also inefficient since all feature/value pairs of that frame must be checked.

The inefficiencies of the intermediate representation is recovered by the second phase which takes this representation of TMRs and produces an equivalent but efficient representation of those TMRs in Prolog. This representation utilizes the *predicate-name/first-argument* indexing facility of Prolog and the tripartite $(Frame_{id}, Feature, Value)$ structure of a TMR frame. Since there can be multiple TMRs in the input, the distinction between those TMRs should also be achieved. So, the tripartite structure of a TMR frame is represented by the following Prolog program:

$$\begin{aligned} &clause_i(Frame_i, Feature_1, Value_1). \\ &\qquad\vdots \\ &clause_i(Frame_i, Feature_n, Value_n). \end{aligned}$$

Although the same representation can be used for relation frames (temporal relations, domain relations, etc.), since their content provides an unit information, they are produced in the following way:

$$\begin{aligned} &clause_i(Relation_i, type, Value). \\ &clause_i(Relation_i, arg_1, Frame_i). \\ &clause_i(Relation_i, arg_2, Frame_j). \\ &\qquad\Downarrow \\ &clause_i(Relation_i, RelationType(Frame_i, Frame_j)). \end{aligned}$$

While the second phase is processing the frames in a TMR, it constructs three lists that contain the event frames, instantiated concept frames, and parent/children relationships. This is done to avoid the inefficiencies arising from the intermediate representation. The event list is extracted directly from *table-of-contents* frame. The instantiated concept frame list and the parent/children relationship list are updated at every frame that is processed. These lists are represented by following Prolog program in which predicate index has the same value as the clause index.

$$\begin{aligned} &event_i(Frame_i). \\ &concept_i(Frame_i). \\ &relation_i(Frame_i, Frame_j). \end{aligned}$$

The overall architecture of the implemented TMR Parser is shown in the Figure 5.1.

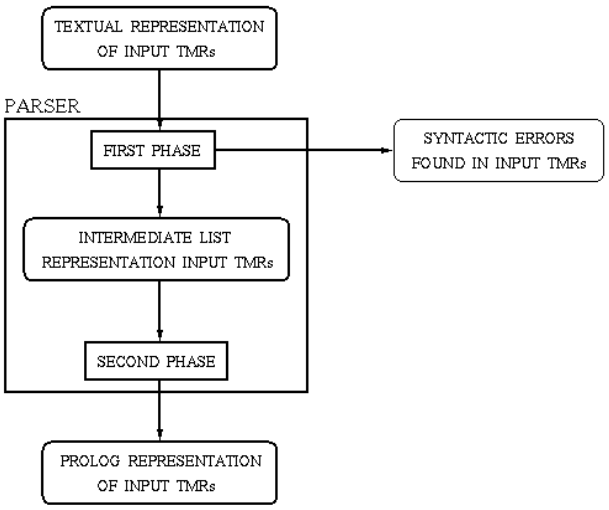

Figure 5.1: Architecture of the TMR Parser

5.2 Representation of Knowledge Resources

Since knowledge resources are at the core of the developed system and the information which resides in them is retrieved all the time during the processing of an input TMR, it is very important to design an efficient representation for them in Prolog. The property of Prolog, *predicate-name/first-argument* indexing, is utilized in the design of knowledge resources like in the Prolog representation of a TMR which improves the retrieval complexity a lot.

A concept in the *ontology* is composed of four components as described in Chapter 3. The first component is its parent concepts and they are introduced through *is-a* feature. The second one provides the allowed thematic roles and the possible values those roles can take, and they are introduced through *roles* feature. *Roles* feature is represented as a list of *role-name/role-value-list* pairs. In the case of no specific role-value requirement, *concept* is given as the content of the *role-value-list*. The third one defines the decompositional properties of that concept with the domains of those properties by the *definition* feature. *Definition* is represented as a list of *feature-name/feature- domain* pairs. As explained in Chapter 4, there are three types of domains and they are represented by *unordered(domain-name,domain-size)*, *ordered(domain-name,domain-size)*, and *numeric(start-point,end-point,domain-size,incremental-unit)*. *Domain-name* provides the allowed values for that feature. The final component is introduced if that concept has some relationships with other concepts in the ontology such as *is-part-of*, *made-of*, etc. So, the following is

an imaginary concept definition in the ontology:

$$
\begin{array}{l}
concept\text{-}name(is\text{-}a, parent_1). \\
\qquad\qquad \vdots \\
concept\text{-}name(is\text{-}a, parent_n). \\
\\
concept\text{-}name(roles, \\
\qquad\qquad [[role_1, [concept_i, \ldots, concept_j]], \\
\qquad\qquad\qquad\qquad \vdots \\
\qquad\qquad [role_n, [concept_i, \ldots, concept_j]]]). \\
\\
concept\text{-}name(definition, \\
\qquad\qquad [[feature_1, unordered(domain_1, size_1)], \\
\qquad\qquad [feature_2, ordered(domain_2, size_2)], \\
\qquad\qquad [feature_3, numeric(begin_1, end_1, size_3, unit_1)], \\
\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\qquad\qquad [feature_n, numeric(begin_m, end_m, size_n, unit_m)]]). \\
\\
\begin{array}{lll}
domain_1(value_1). & domain_2(value_1). & \\
\quad \vdots & \quad \vdots & \ldots \\
domain_1(value_n). & domain_2(value_m). &
\end{array} \\
\\
concept\text{-}name(relation_1, concept_i). \\
\qquad\qquad \vdots \\
concept\text{-}name(relation_n, concept_j).
\end{array}
$$

A lexeme entry in the *lexicon* is composed of four components, like a concept entry in the ontology. Two of the components are same, *roles* and *definition*, although their representations are a little bit different. First, the definition of a lexeme divides the allowed thematic roles into two groups, *required* and *optional* that are explained in Chapter 3 and this division must be introduced. Second, instead of defining the domain of a feature, its allowed range of values with its *importance* value should be provided. The third component provides the lexeme's categorical information in *category*. Last component is used if that lexeme requires some pragmatic context in order to be used and this is given in *pragmatic*. Also the concept which is used in the definition of the lexeme should be provided. So, the following is an imaginary lexeme definition in the lexicon:

$$concept\text{-}name(language, lexeme_i).$$

$$lexeme_i(category, language,$$
$$[feature(category, Category), feature(root, lexeme)]).$$

$$lexeme_i(roles, language,$$
$$[required([[role_1, [concept_i, \ldots, concept_j]],$$
$$\vdots$$
$$[role_k, [concept_i, \ldots, concept_j]]]),$$
$$optional([[role_l, [concept_i, \ldots, concept_j]],$$
$$\vdots$$
$$[role_m, [concept_i, \ldots, concept_j]]])]).$$

$$lexeme_i(definition, language,$$
$$[feature(name_1, value_1, importance_1),$$
$$\vdots$$
$$feature(name_m, value_m, importance_m)]).$$

$$lexeme_i(pragmatics, language,$$
$$[stylistics([[color, value_1], \ldots]),$$
$$attitude([[type, attitude\text{-}type], \ldots])]).$$

All applicable *map-rules* for a frame type or a concept are grouped in a unique set through using the *any* property which is described in Chapter 3. This set comprises the set of independent rules and the rules that exclude the application of some others are grouped under *exclusive* property. Since all of the rules in that set should be applied to the input TMR, they can be represented as a list of rules without degrading the efficiency. So, the map-rules associated with an imaginary TMR type is represented in Prolog by the following format:

$$maprule(type, language,$$
$$rule(any, [rule(exclusive,$$
$$[[conditions_1, updates_1],$$
$$\vdots$$
$$[conditions_k, updates_k]]),$$
$$\vdots$$
$$rule(exclusive,$$
$$[[conditions_1, updates_1],$$
$$\vdots$$
$$[conditions_m, updates_m]]),$$
$$[conditions_1, updates_1],$$
$$\vdots$$
$$[conditions_n, updates_n]])).$$

As explained in Chapter 4, to improve efficiency of the tree representation of *f-structure* is converted into one level hierarchic representation that is utilized in ontology. Remember that, there are two kinds of construction in an f-structure tree: *slots* and *features*. This distinction is provided through an argument and the children slots of a slot form the parent-child relationships in the tree representation. The domain of a feature is defined like in the ontology and the roots of available trees are given by *top-syntax* features. So, the following is an imaginary syntax definition of a specific language's f-structure:

$$
\begin{array}{l}
top\text{-}syntax(root_1).\\
\quad\vdots\\
top\text{-}syntax(root_k).\\
\\
syntax(root_1, feature_1, feature, domain_1).\\
\quad\vdots\\
syntax(root_1, feature_m, feature, domain_m).\\
syntax(root_1, slot_1, slot, slot\text{-}name_1).\\
\quad\vdots\\
syntax(root_1, slot_n, slot, slot\text{-}name_n).\\
domain_1(value_1). \qquad\qquad domain_m(value_1).\\
\quad\vdots \qquad\qquad\qquad\qquad \vdots\\
domain_1(value_i). \qquad\qquad domain_m(value_j).\\
\\
\qquad\qquad\qquad\vdots\\
\\
syntax(root_k, feature_1, feature, domain_p).\\
\quad\vdots\\
syntax(root_k, slot_1, slot, slot_r).\\
\\
\qquad\qquad\qquad\vdots
\end{array}
$$

5.3 Time Complexity of the System

Since the computational model described in Chapter 4 is a knowledge processing system and the complexity of knowledge requests and updates is dependent on various aspects, its time complexity cannot be given in exact mathematical notations. So, in this section, only the aspects that have an effect on the time complexity of the system are explained with some approximations about their complexities. First, note that all of the followings affect the processing time of the implemented system:

- Complexity of each lexical selection request
- Complexity of meaning requirement checks defined in map-rules
- Complexity of each f-structure update operation request
- Overall complexity of the input TMR

The complexity of a lexical selection request depends on the content of the TMR frame which activates the module and the information available in the lexicon. In fact, there are six independent properties of these two resources that affect the overall complexity of the lexical selection module and they are listed below:

- Number of lexemes that are available in the lexicon which are defined by the current TMR frame's concept
- The complexity of the *thematic roles* definitions of the candidates which are checked in context-dependent selection
- Number of candidates that are eliminated by the context-dependent selection which affects the complexity of context-independent selection
- Number of features used as *definition* for both the TMR frame and the candidate lexemes, since all of them should be checked
- Complexity of values in *definition* features since calculation requirements of a *single*, *enumerated*, and *range* filler is different
- Need to use *pragmatic* information because of existing ambiguity in the selection after context-independent selection

So, the complexity of lexical selection module can be calculated in the following way, which is overtly depends on average values that are changed in every update of the lexicon.

n = average number of lexemes found in the lexicon
m = average number of thematic role requirements for a lexeme
α = average proportion of elimination in context-dependent selection
p = average number of features used as definition
β = average proportion of single fillers
γ = average proportion of enumerated fillers
l = average length of an enumerated filler
ρ = average proportion of range fillers
μ = average proportion of using pragmatic information
$Comp_{prag}$ = average complexity of pragmatic processing

$$
\begin{aligned}
Comp_{lex\text{-}sel} = \; & n * m + \\
& \alpha * \beta * p * n + \\
& \alpha * \gamma * l * p * n + \\
& \alpha * \rho * p * n + \\
& \mu * \alpha * n * Comp_{prag}
\end{aligned}
$$

The complexity of a map-rule application, which is associated with a TMR frame type, depends on that map-rule's internal complexity. There are four internal properties of a map-rule that have an effect on the overall complexity of a single map-rule application, and they are listed below:

- Number of independent rules that are grouped under that map-rule by using *any* type, since each of them should be checked individually,
- Number of *exclusive* rules in that *any* group, their internal complexity, (number of rules grouped under those *exclusive* set), and the success rate of each individual rule in that group,
- Number of references required for checking the content of the input TMR and their internal complexity. For example, referring to a child frame's content of a TMR frame introduces more processing compared with referring to the content of that TMR frame,
- Number of f-structure update requests made by that rule and the depth of the feature in the f-structure tree to be inserted (*map* requests does not introduce any time complexity).

So, the complexity of a single map-rule application can be calculated in the following way which heavily relies on average values, like the previous calculation.

n = average number of independent rules grouped under *any*
α = average proportion of *exclusive* rules
m = average number of dependent rules grouped under *exclusive*
β = average proportion for reaching a successful rule in *exclusive*
γ = average proportion for declaring a rule as failed
p = average number of references made in an individual rule
$Comp_{ref}$ = average internal complexity of a reference to TMR
r = average number of f-structure update operations in an individual rule
d = average depth of a feature in f-structure tree to be inserted

$$
\begin{aligned}
Comp_{any} = \; & \gamma * p * Comp_{ref} + \\
& (1 - \gamma) * p * Comp_{ref} + r * d
\end{aligned}
$$

$$Comp_{exclusive} = \gamma * p * Comp_{ref} * \beta * m + p * Comp_{ref} + r * d$$

$$Comp_{map\text{-}rule} = \alpha * n * Comp_{exclusive} + (1 - \alpha) * n * Comp_{any}$$

The overall complexity of processing an input TMR is heavily depends on the complexity of the TMR itself. In fact, there are two things that determine the processing time of a TMR: the number of frames that reside in the TMR, and the number of frames that are instantiated concepts. Since processing of an instantiated concept activates lexical selection module and causes the application of all map-rules that are associated with the concept's ancestors, the proportion of instantiated concepts over the number of frames in a TMR have a big effect in the complete complexity. Note that, the depth of the concept in the ontology and the average number of parents of a concept (the structure of the ontology) also affects the processing complexity. So, the complexity of the overall system can be explained by the following calculation:

n = average number of frames in an input TMR
α = average proportion of instantiated concepts in TMR
d = average depth of a concept in the ontology
β = average number of parent concepts of a concept in the ontology

$$Comp_{processing} = \alpha * n * Comp_{lex\text{-}sel} + \alpha * n * Comp_{map\text{-}rule} + \alpha * n * \sum_{i=0}^{d} (\beta^i * Comp_{map\text{-}rule}) + (1 - \alpha) * n * Comp_{map\text{-}rule}$$

Chapter 6

Conclusion and Future Work

The goal of this work is to develop and implement a language-independent system that takes the meaning representation of a sentence (TMR) [3, 18] as input and produces the feature structure representation of that sentence in a target language. To achieve such a task, the system utilizes four knowledge resources. The first knowledge resource, *ontology*, provides the abstract representation of the world and it is utilized in the meaning representation. The other three knowledge resources provide information about the target language which are *lexicon* (word knowledge), *map-rules* (structural mapping between meaning and f-structure representation), and the *feature structure* representation of the target language. By using these knowledge resources and processing the input TMR, the system selects lexical items and constructs the syntactic structure of the output sentence.

Although the general structure of the system is taken from [10, 22], some components described in the previous chapters are redesigned. First, the structure of map-rules described in Chapter 3 is designed in this work. The proposed method for checking the content of a TMR, meaning requirements of a map-rule, are both efficient and modular. Also, with that method, the design of ad-hoc rules is avoided which is one of the corner stones of interlingua methodology. Second, the efficient and the general design of the feature structure representation is also developed in this thesis. The algorithm for performing f-structure update operations, which is described in Chapter 4, is also designed in this work. Third, the order of frame processing (depth-first) is proposed and utilized in processing TMRs that have more than one event inside. The method for making connection between the events of a TMR (in Chapter 4) is also proposed and implemented in the developed system.

The developed system is implemented in Prolog. Although the representa-

tion utilized for TMR is user friendly and easy to create, it is very difficult for processing by a programming language. So, a new representation is developed that can be efficiently used in Prolog. Also, to avoid losing that user friendly representation of TMR, a new tool, TMR Parser, is implemented that takes TMRs from a text file and produces their Prolog representations with a report about the possible mistakes encountered. Prolog representations of knowledge resources are efficient in knowledge retrieval and modular.

The system that is developed can be used to produce the syntactic structure of a language from the abstract meaning representation (TMR). This syntactic structure can be then fed into the tactical generator of that language to achieve generation of sentences in that language from TMRs. To process a TMR in a language, only the knowledge resources should be developed without interfacing with the system itself. Also, TMR Parser allows for checking semantic and pragmatic phenomenon in a language without waiting for a parser to produce the text meaning representation given as input to the system.

The implemented system is tested with Turkish. But, since developing such a system is not an easy job, the sizes of the knowledge resources , *lexicon* and *map-rules*, are very small. The contents of these knowledge resources are generally developed to test the specific components of the developed system. So, currently the system is far from covering Turkish lexical items and syntactic constructions used for denoting semantic and pragmatic phenomenon. Many lexical items should be added to Turkish lexicon and Turkish map-rules should be redeveloped and made richer with a deep analysis of Turkish sentences. Only with these developments, a real generation system for Turkish can be produced with Hakkani's tactical generator [11].

There is also some future work if the described and implemented system is considered. First of all, connecting events that are not the main event of a TMR is not handled by the current system. The algorithm that is used for relating events (in Chapter 4) should be revised to cover these cases. Secondly, currently available meaning requirement check methods may not be enough and new ones should be designed with a new analysis of languages. Thirdly, although the current *parser* covers most of the syntax of TMR, it still needs some refinement to work properly for any TMR.

References

[1] D. Arnold and L. Tombe. *Basic Theory and Methodology in EUROTRA*. Cambrige University Press, Cambridge, 1987.

[2] J. R. Bateman. Ontology construction and natural language. In *Proceedings of International Workshop on Formal Ontology*, pages 83–93, Padua, Italy, 1993.

[3] S. Beale, S. Nirenburg, and K. Mahesh. Semantic analysis in the mikrokosmos machine translation project. In *Proceedings of the 2nd Symposium on Natural Language Processing (SNLP-95)*, Bangkok, Thailand, August 2-4, 1995.

[4] W. S. Benneth. The linguistic components of metal. In *Working Paper, Linguistic Research Center*, University of Texas, Austin, 1982.

[5] B. Comrie. *Aspect*. Cambridge University Press, Cambridge, 1976, 1991.

[6] B. Comrie. *Tense*. Cambridge University Press, Cambridge, 1985, 1990.

[7] B. J. Door. The use of lexical semantics in interlingua machine translation. *Machine Translation*, 4:3:135–193, 1993.

[8] D. Frawell and Y. Wilks. Ultra: A multi-lingual machine translator. In *Proceedings of Machine Translation Summit III*, Washington, DC, 1991.

[9] W. Frawley. *Linguistic Semantics*. Lawrance Erlbaum Associates, Hillsdale, New Jersey, 1992.

[10] K. Goodman and S. Nirenburg. *The KBMT Project: A Case Study in Knowledge-Based Machine Translation*. Morgan Kaufmann, San Mateo, California, 1991.

[11] D. Z. Hakkani. Design and implementation of a tactical generator for turkish, a free constituent order language. Master's thesis, Bilkent University, Ankara Turkey, July 1996.

[12] W. J. Hutchins. *Machine Translation: Past, Present, Future*. Ellis Horword Ltd., Chichester, U.K., 1986.

[13] W. J. Hutchins and H. L. Somers. *An Introduction to Machine Translation*. Academic Press, London, U.K., 1992.

[14] J. R. Leavit, D. Lonsdale, H. Keck, and Nyberg E. Tooling the lexicon acquisition process for large-scaled knowledge-based machine translation. In *Proceedings of IEEE Tools for AI*, 1994.

[15] J. R. Leavitt, Lonsdale D., and A. Franz. A reasoned interlingua for knowledge-based machine translation. In *Proceedings of Canadian Artificial Intelligence Conference*, Banff, Canada, 1994.

[16] K. Mahesh. Ontology development for machine translation: Ideology and methodology. In *Memoranda in Computer and Cognitive Science MCCS-96-292*, Las Crues, New Mexico State University, 1996.

[17] K. Mahesh and S. Nirenburg. A situated ontology for practical nlp. In *Proceedings of Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence*, Montreal, Canada, August 19-20, 1995.

[18] K. Mahesh and S. Nirenburg. Meaning representation for knowledge sharing in practical machine translation. In *Proceedings of the FLAIRS-96. Track on Information Interchange, Florida AI Research Symposium*, Key West, Florida, May 19-22, 1996.

[19] K. Mahesh and S. Nirenburg. Semantic classification for practical natural language processing. In *Proceedings of Sixth ASIS SIG/CR Classification Research Workshop: An Interdisciplinary Meeting*, Chicago IL, October 8, 1995.

[20] T. Mitamura and E. Nyberg. Hierarchical lexical structure and interpretive mapping in machine translation. In *Proceedings of COLING-92*, Nantes, France, July, 1992.

[21] T. Mitamura and E. Nyberg. Controlled english for knowledge-based mt: Experience with the kant system. In *Proceedings of 6th International Conference on Theoretical and Methodological Issues in Machine Translation*, Leuven, Belgium, July 5-7, 1995.

[22] S. Nirenburg, J. Carbonell, M. Tomita, and K. Goodman. *Machine Translation: A Knowledge-Based Approach.* Morgan Kaufmann, San Mateo, California, 1992.

[23] S. Nirenburg and C. Defrise. Application-oriented computational semantics. In *Computational Linguistic and Formal Semantics. In Johnson, R. and Rosner, M., editors*, pages 223–256, Cambridge University Press, 1994.

[24] S. Nirenburg and K. Goodman. Treatment of meaning in mt systems. In *Proceedings of Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, pages 171–188, University of Texas, Austin, 1990.

[25] S. Nirenburg and V. Raskin. The analysis lexicon and the lexicon management. *Computers and Translation*, 2:177–188, 1987.

[26] E. Nyberg and T. Mitamura. The kant system: Fast, accurate, high-quality translation in practical domains. In *In Proceedings of COLING-92*, Nantes, France, July, 1992.

[27] F. R. Palmer. *Mood and Modality.* Cambridge University Press, Cambridge, 1986, 1991.

[28] Korkmaz T. Turkish text generation with systemic-functional grammar. Master's thesis, Bilkent University, Ankara Turkey, June 1996.

[29] M. Vasconcellos and M. Leon. Spanam and engspan: Machine translation at the pan american health organization (paho). *Computational Linguistics*, 11:122–136, 1985.

[30] P. Wheeler. Changes and improvements to the european commission's systran system. In *Proceedings of the International Conference on the Methodologies and Techniques of Machine Translation*, Cranfield Institute of Technology, U.K., 1984.

Appendix A

A Sample Run of the TMR Parser

To show how the implemented TMR Parser is utilized in finding the mistakes that are made in the manual creation of input TMRs and constructing the real input format of TMRs, a simple example is given in this section. This example presents a simple TMR created for the sentence "Ali went to the school" in which there are some deliberate mistakes.

table-of-contents$_1$

speech-act	*speech-act*$_1$
heads	%*go*$_1$
time-rels	*temp-rel*$_1$

%*go*$_1$

agent	%*human*$_1$
destination	%*location*$_1$
polarity	*positive*
aspect	*aspect*$_1$
times	*time*$_1$

aspect$_1$

phase	*perfect*
iteration	*single*
duration	*prolonged*
telicity	*tru*

speech-act$_1$

type	*declarative*
scope	%*go*$_1$
time	*time*$_2$

%*humann*$_1$

type	*proper*
name	*'Ali'*

%*location*$_1$

type	*school*
reference	*definite*

time$_1$

absolute	*past*

time$_3$

absolute	*past*

temp-rel$_1$

type	*after*
arg$_1$	*time*$_2$
arg$_2$	*time*$_1$

The first phase of the *parser* gets this textual TMR and produces the following list of mistakes that are encountered in that TMR.

ERROR !!!
  FEATURE = *times* is not found in DEFINITION of $\%go_1$
ERROR !!!
  FEATURE = *time* is required by DEFINITION of $\%go_1$
ERROR !!!
  FRAME = $\%humann_1$ is not found in ONTOLOGY
ERROR !!!
  VALUE = *tru* is not a valid value in DEFINITION of $aspect_1$
ERROR !!!
  FRAME = $\%human_1$ is referenced in $\%go_1$, but not defined
ERROR !!!
  FRAME = $time_2$ is referenced in $speech\text{-}act_1$, but not defined
ERROR !!!
  FRAME = $time_3$ is defined, but not referenced in the TMR

After correcting the mistakes found in the TMR, the new input is parsed again by the first phase and the following intermediate list representation of the TMR is produced.

Intermediate-List-Representation =

$[\,[\,[type, tc], [id, 1], [heads, [[go, 1]]], [temp\text{-}rels, [1]]]$
$\;[\,[type, instantiated], [name, go], [id, 1], [agent, [human, 1]],$
$\;\;[destination, [location, 1]], [polarity, positive], [aspect, 1], [time, 1]]$
$\;[\,[type, instantiated], [name, human], [id, 1], [type, proper], [name, `Ali']]$
$\;[\,[type, instantiated], [name, location], [id, 1],$
$\;\;[type, school], [reference, definite]]$
$\;[\,[type, aspect], [id, 1], [phase, perfect],$
$\;\;[iteration, single], [duration, prolonged], [telicity, true]]$
$\;[\,[type, time], [id, 1], [absolute, past]]$
$\;[\,[type, speech\text{-}act], [id, 1], [type, declarative], [scope, [go, 1]], [time, 2]]$
$\;[\,[type, time], [id, 2], [absolute, past]]$
$\;[\,[type, temp\text{-}rel], [id, 1], [type, after], [arg_1, 2], [arg_2, 1]]]$

The second phase processes the list above and produces the following Prolog program which is the real input to the computational model described in Chapter 4.

$clause1(speech\text{-}act, inst(speech\text{-}act, 1)).$
$clause1(temp\text{-}rel, inst(temp\text{-}rel, 1)).$

$clause1(inst(go, 1), agent, inst(human, 1)).$

$clause1(inst(go, 1), destination, inst(location, 1)).$
$clause1(inst(go, 1), polarity, positive)).$
$clause1(inst(go, 1), aspect, inst(aspect, 1)).$
$clause1(inst(go, 1), time, inst(time, 1)).$

$clause1(inst(human, 1), type, proper)).$
$clause1(inst(human, 1), name, `Ali')).$

$clause1(inst(location, 1), type, school).$
$clause1(inst(location, 1), reference, definite).$

$clause1(inst(aspect, 1), phase, perfect).$
$clause1(inst(aspect, 1), iteration, single).$
$clause1(inst(aspect, 1), duration, prolonged).$
$clause1(inst(aspect, 1), telicity, true).$

$clause1(inst(time, 1), absolute, past).$

$clause1(inst(speech\text{-}act, 1), type, declarative).$
$clause1(inst(speech\text{-}act, 1), scope, inst(go, 1)).$
$clause1(inst(speech\text{-}act, 1), time, inst(time, 2)).$

$clause1(inst(time, 2), absolute, past).$
$clause1(inst(temp\text{-}rel, 1), after(inst(time, 2), inst(time, 1))).$

$head1(inst(go, 1)).$

$concept1(inst(go, 1)).$
$concept1(inst(human, 1)).$
$concept1(inst(location, 1).$

$relation1(clause1, inst(speech\text{-}act, 1)).$
$relation1(clause1, inst(temp\text{-}rel, 1)).$
$relation1(inst(go, 1), inst(human, 1)).$
$relation1(inst(go, 1), inst(location, 1)).$
$relation1(inst(go, 1), inst(aspect, 1)).$
$relation1(inst(go, 1), inst(time, 1)).$
$relation1(inst(speech\text{-}act, inst(go, 1)).$
$relation1(inst(speech\text{-}act, inst(time, 2)).$

Appendix B

A Trace of the Model

To explain, once more, how the computational model described in Chapter 4 processes an input TMR, the following example is presented. In this example, the *trace* facility of the implemented Prolog program is used instead of manual writing. The input TMR is created for the Turkish sentence "Ali camı kıracaktı.", which can be translated into English as "Ali would have broken the window". This sentence expresses an expectation about the event, which did not occur ($modality_2$), in the past ($modality_1$).

*table-of*contents

speech-act	$speech\text{-}act_1$
heads	$BREAK_1$
temp-rels	$temp\text{-}rel_1$
modalities	$modality_1$, $modality_2$

$BREAK_1$

agent	$HUMAN_1$
patient	$WINDOW_1$
polarity	*positive*
time	$time_1$
modality	$modality_1$

$HUMAN_1$

type	*proper*
name	*ali*

$WINDOW_1$

reference	*definite*

$modality_1$

type	*expectative*
value	1
scope	$BREAK_1$
attribution	*speaker*

$speech\text{-}act_1$

type	*declarative*
scope	$BREAK_1$
time	$time_2$
modality	$modality_2$

$time_2$

absolute	*past*

$aspect_1$		$modality_2$	
phase	*begin*	*type*	*epistemic*
iteration	*single*	*value*	0
duration	*momentary*	*scope*	$BREAK_1$
telicity	*false*	*attribution*	*speaker*
$time_1$		$temp\text{-}rel_1$	
absolute	*past*	*type*	*after*
		arg_1	$time_2$
		arg_2	$time_1$

Giving this TMR as an input to the implemented system with the *trace* facility activated, the following output is produced. To save space, some of the failed rules are not shown.

```
START PROCESSING FRAME = inst(break,1)
   LEXICAL SELECTION
      Found Lexicals for Concept = break
         [kIr1]
      Only one Lexeme defined for:
         CONCEPT = break
         LANGUAGE = turkish
         LEXEME = kIr1
   END (LEXICAL SELECTION) !!!
   Selected Lexeme = kIr1

   MAP-RULE APPLICATION (CONCEPT) !!!
      feature(category,verb) is added
      feature(root,kIr) is added
      F-Structure Updated

      Applying Any-Rule
         Found = exist(processing,agent)
         Rule Succeeded !!!
         map(agent,subject) is applied
         F-Structure Updated

         Found = exist(processing,patient)
         Rule Succeeded !!!
         map(patient,dir-object) is applied
         F-Structure Updated
      Any-Rule Applied

      Applying Any-Rule
         Applying Exclusive-Rule
            Rule Failed !!!
               Not Found = head(hear)
               Not Checked =
```

```
                ref(context,theme,event)
                time(after,[speech,context,event])
                modality(speech,[epistemic,lse(0.75),speaker,event])
                aspect(event,[perfect,iterative,momentary,true])

            Rule Failed !!!
               Not Found = aspect(event,[perfect,iterative,momentary,true])
               Not Checked =
                  time(after,[speech,event])

            Rule Failed !!!
               Not Found = aspect(event,[continue,iterative,prolonged,true])
               Not Checked =
                  time(after,[speech,event])

            Found = time(after,[speech,event])
            Found = modality(event,[expectative,eq(1),speaker,event])
            Found = modality(speech,[epistemic,eq(0),speaker,event])
            Found = aspect(event,[begin,single,momentary,false])
            Rule Succeeded !!!
            feature(tense,future) is added
            feature(mode,past) is added
            F-Structure Updated
         Exclusive-Rule Applied
      Any-Rule Applied

      Applying Any-Rule
         Found = ref(processing,polarity,positive)
         Rule Succeeded !!!
         feature(sense,positive) is added
         F-Structure Updated

         Found = ref(speech,scope,processing)
         Rule Succeeded !!!
         feature(clause-type,predicative) is added
         F-Structure Updated
      Any-Rule Applied

      Applying Any-Rule
         Found = same(processing,event)
         Rule Succeeded !!!
         feature(s-form,finite) is added
         F-Structure Updated
      Any-Rule Applied
   END (MAP-RULE APPLICATION) !!!

   Indirectly Processed !!! (inst(aspect,1))
   Indirectly Processed !!! (inst(modality,1))
```

```
PROCESSED !!! (inst(break,1))

F-STRUCTURE =
   [[s-form,finite],[clause-type,predicative],
    [verb,[[sense,positive],[mode,past],[tense,future],[root,kIr],[category,verb]]]]

START PROCESSING FRAME = inst(human,1)
   LEXICAL SELECTION
      Found Lexicals for Concept = human
         [adam1,kadIn1,Cocuk1,Ali1]
      Starting CONTEXT-DEPENDENT Selection
         Lexical = adam1 OK !!!
         Lexical = kadIn1 OK !!!
         Lexical = Cocuk1 OK !!!
         Lexical = Ali1 OK !!!
      End of CONTEXT-DEPENDENT Selection
      Remained Lexemes = [Ali1,Cocuk1,kadIn1,adam1]
      Starting CONTEXT-INDEPENDENT Selection
         Lexical/Penalty = Ali1/0 !!!
         Lexical/Penalty = Cocuk1/19 !!!
         Lexical/Penalty = kadIn1/27 !!!
         Lexical/Penalty = adam1/27 !!!
      End of CONTEXT-INDEPENDENT Selection
   END (LEXICAL SELECTION) !!!
   Selected Lexeme = Ali1

   MAP-RULE APPLICATION (CONCEPT) !!!
         feature(category,noun) is added
         feature(root,Ali) is added
         F-Structure Updated

      Applying Any-Rule
         Found = ref(processing,type,proper)
         Rule Succeeded !!!
         feature(number,singular) is added
         feature(person,third) is added
         F-Structure Updated
      Any-Rule Applied

      Applying Any-Rule
         Rule Failed !!!
            Not Found = ref(speech,scope,processing)
            Not Checked =

         Rule Failed !!!
            Not Found = ref(processing,reference,definite)
            Not Checked =
      Any-Rule Applied
```

```
        Applying Any-Rule
          Rule Failed !!!
            Not Found = same(processing,event)
            Not Checked =
        Any-Rule Applied
    END (MAP-RULE APPLICATION) !!!
PROCESSED !!! (inst(human,1))

F-STRUCTURE =
   [[s-form,finite],[clause-type,predicative],
    [verb,[[sense,positive],[mode,past],[tense,future],[root,kIr],[category,verb]]],
    [arguments,
     [[subject,[[referent,[[arg,[[root,Ali],[category,noun]]],
                          [agr,[[person,third],[number,singular]]]]]]]]]

START PROCESSING FRAME = inst(window,1)
   LEXICAL SELECTION
      Found Lexicals for Concept = window
        [cam1]
      Only one Lexeme defined for:
        CONCEPT = window
        LANGUAGE = turkish
        LEXEME = cam1
   END (LEXICAL SELECTION) !!!
   Selected Lexeme = cam1

   MAP-RULE APPLICATION (CONCEPT) !!!
        feature(category,noun) is added
        feature(root,cam) is added
        F-Structure Updated

      Applying Any-Rule
        Rule Succeeded !!!
        feature(number,singular) is added
        feature(person,third) is added
        F-Structure Updated
      Any-Rule Applied

      Applying Any-Rule
        Rule Failed !!!
          Not Found = ref(speech,scope,processing)
          Not Checked =

        Found = ref(processing,reference,definite)
        Rule Succeeded !!!
        feature(definite,positive) is added
        F-Structure Updated
      Any-Rule Applied
```

```
      Applying Any-Rule
        Rule Failed !!!
          Not Found = same(processing,event)
          Not Checked =
      Any-Rule Applied
   END (MAP-RULE APPLICATION) !!!
PROCESSED !!! (inst(window,1))

F-STRUCTURE =
   [[s-form,finite],[clause-type,predicative],
    [verb,[[sense,positive],[mode,past],[tense,future],[root,kIr],[category,verb]]],
    [arguments,
     [[subject,[[referent,[[arg,[[root,Ali],[category,noun]]],
                          [agr,[[person,third],[number,singular]]]]]],
      [dir-object,[[referent,[[arg,[[root,cam],[category,noun]]],
                             [agr,[[person,third],[number,singular]]]]],
                   [specifier,[[quan,[[definite,positive]]]]]]]]]]

START PROCESSING FRAME = inst(speech-act,1)
   MAP-RULE APPLICATION (FRAME) !!!
      Applying Any-Rule
        Found = notexist(processing,focus)
        Rule Succeeded !!!
        feature(voice,active) is added
        F-Structure Updated

        Found = ref(processing,type,declarative)
        Rule Succeeded !!!
        feature(speech-act,declarative) is added
        F-Structure Updated
      Any-Rule Applied
   END (MAP-RULE APPLICATION) !!!

   Indirectly Processed !!! (inst(modality,2))
PROCESSED !!! (inst(speech-act,1))

F-STRUCTURE =
   [[s-form,finite],[clause-type,predicative],[speech-act,declarative],[voice,active],
    [verb,[[sense,positive],[mode,past],[tense,future],[root,kIr],[category,verb]]],
    [arguments,
     [[subject,[[referent,[[arg,[[root,Ali],[category,noun]]],
                          [agr,[[person,third],[number,singular]]]]]],
      [dir-object,[[referent,[[arg,[[root,cam],[category,noun]]],
                             [agr,[[person,third],[number,singular]]]]],
                   [specifier,[[quan,[[definite,positive]]]]]]]]]]
```

Appendix C

Sample TMRs & F-Structures

In this section, some more TMR examples fro Turkish sentences are given to show how the frames and the ontology are utilized to represent the meaning inside an expression. Each sentence presented below is given to show different structures used in TMR. The feature structure representations given for these examples are produced by the developed system (only the output format is changed).

Example 1:

The first sentence is given to demonstrate a simple sentence representation. Note that, the event that is described by that sentence is *punctual* ($aspect_1$).

"Adam kadına bir kitap verdi"
"The man gave an apple to the woman"

table-of-contents	
speech-act	$speech\text{-}act_1$
heads	$GIVE_1$
temp-rels	$temp\text{-}rel_1$

$GIVE_1$	
agent	$HUMAN_1$
theme	$BOOK_1$
goal	$HUMAN_2$
polarity	*positive*
aspect	$aspect_1$
time	$time_1$

$HUMAN_1$	
type	*common*
gender	*male*
age	≥ 18
reference	*definite*

$BOOK_1$	
reference	*indefinite*

$aspect_1$	
phase	*perfect*
duration	*momentary*
iteration	*single*
telicity	*false*

$speech\text{-}act_1$	
type	*declarative*
scope	$GIVE_1$
time	$time_2$

$HUMAN_2$		$time_2$	
type	*common*	*absolute*	*past*
gender	*female*		
age	≥ 18	$temp\text{-}rel_1$	
reference	*definite*	*type*	*after*
		arg_1	$time_2$
$time_1$		arg_2	$time_1$
absolute	*past*		

F-Structure =
[[s-form,finite], [clause-type,predicative], [speech-act,declarative], [voice,active],
[verb, [[sense,positive], [mode,past], [root,'ver'], [category,verb]]],
[arguments,
[[subject, [[referent, [[arg, [[root,'adam'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan,[[definite,positive]]]]]],
[dir-object, [[referent, [[arg, [[root,'kadın'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan, [[definite,positive]]]]]],
[beneficiary, [[referent, [[arg, [[root,'kitap'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]]]]]

Example 2:

The second sentence is given to show how word ordering phenomena in Turkish can be represented by a TMR. Note that, the salient argument, $HUMAN_1$, is updated with $attitude_1$. Also, the argument which is given as a background (it should not be mentioned), $BOOK_1$, is marked with $attitude_2$.

"Kadına o adam verdi kitabı"

"It was that man who gave something, the book, to the woman"

table-of-contents	
speech-act	$speech\text{-}act_1$
heads	$GIVE_1$
temp-rels	$temp\text{-}rel_1$
attitudes	$attitude_1$, $attitude_2$

$GIVE_1$		$BOOK_1$	
agent	$HUMAN_1$	*reference*	*definite*
theme	$BOOK_1$	*attitude*	$attitude_2$
goal	$HUMAN_2$		
polarity	*positive*	$attitude_2$	
aspect	$aspect_1$	*type*	*saliency*
time	$time_1$	*value*	≤ 0.25
		scope	$book_1$
		attribution	*speaker*

$HUMAN_1$	
type	*common*
gender	*male*
age	≥ 18
reference	*definite*
distance	*far*
attitude	$attitude_1$

$attitude_1$	
type	*saliency*
value	≥ 0.75
scope	$HUMAN_1$
attribution	*speaker*

$HUMAN_2$	
type	*common*
gender	*female*
age	≥ 18
reference	*definite*

$aspect_1$	
phase	*perfect*
duration	*momentary*
iteration	*single*
telicity	*false*

$speech\text{-}act_1$	
type	*declarative*
scope	$GIVE_1$
time	$time_2$

$time_2$	
absolute	*past*

$temp\text{-}rel_1$	
type	*after*
arg_1	$time_2$
arg_2	$time_1$

$time_1$	
absolute	*past*

F-Structure =

[[s-form,finite], [clause-type,predicative], [speech-act,declarative], [voice,active],
[verb, [[sense,positive], [mode,past], [root,'ver'], [category,verb]]],
[arguments,
[[subject, [[referent,[[arg, [[root,'adam'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan, [[definite,positive]]],
[demons,o]]]]],
[dir-object, [[referent, [[arg, [[root,'kadın'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan, [[definite,positive]]]]]]],
[beneficiary, [[referent, [[arg, [[root,'kitap'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]]]]],
[control, [[topic,beneficiary], [focus,subject], [background,dir-object]]]]

Example 3:

The third sentence is given to show how a passive construction in Turkish can be represented by a TMR. Note that, in the sentence below, the passivization is required because of unknown *agent*.

"Kadına bir kitap verildi"

"A book was given to the woman"

table-of-contents

speech-act	$speech\text{-}act_1$
heads	$GIVE_1$
temp-rels	$temp\text{-}rel_1$

$GIVE_1$

agent	*unknown*
theme	$BOOK_1$
goal	$HUMAN_1$
polarity	*positive*
aspect	$aspect_1$
time	$time_1$

$BOOK_1$

reference	*indefinite*

$HUMAN_1$

type	*common*
gender	*female*
age	≥ 17
reference	*definite*

$time_1$

absolute	*past*

$aspect_1$

phase	*perfect*
duration	*momentary*
iteration	*single*
telicity	*false*

$speech\text{-}act_1$

type	*declarative*
scope	$GIVE_1$
time	$time_2$

$temp\text{-}rel_1$

type	*after*
arg_1	$time_2$
arg_2	$time_1$

F-Structure =

[[s-form,finite], [clause-type,predicative], [speech-act,declarative], [voice,passive],
[verb, [[sense,positive], [mode,past], [root,'ver'], [category,verb]]],
[arguments,
[[dir-object, [[referent, [[arg, [[root,'kadın'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan, [[definite,positive]]]]]],
[beneficiary, [[referent, [[arg, [[root,'kitap'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]]]]]]

Example 4:

The example given below shows how an existential sentence is represented in TMR. Note that, the main event of the representation is a set, which denotes an existential construction.

"Masada üç fizik kitabı vardı"

"There were three physics book on the table"

table-of-contents

speech-act	$speech\text{-}act_1$
heads	set_1
temp-rels	$temp\text{-}rel_1$

set_1

member-type	$BOOK_1$
cardinality	3
locative	$TABLE_1$
polarity	*positive*
aspect	$aspect_1$
time	$time_1$

$BOOK_1$

type	*physics*

$aspect_1$

phase	*perfect*
duration	*prolonged*
iteration	*single*
telicity	*true*

$TABLE_1$

reference	*definite*

$time_1$

absolute	*past*

$speech\text{-}act_1$

type	*declarative*
scope	set_1
time	$time_2$

$time_2$

absolute	*past*

$temp\text{-}rel_1$

type	*after*
arg_1	$time_2$
arg_2	$time_1$

F-Structure =

[[s-form,finite], [clause-type,existential], [speech-act,declarative], [voice,active],
[verb, [[sense,positive], [mode,past], [root,'var'], [category,verb]]],
[arguments,
[[subject, [[referent, [[arg, [[root,'kitap'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[classifier, [[referent, [[arg, [[root,fizik], [category,noun]]]]]]],
[modifier, [[quantifier, [[low,3], [high,nil]]]]]],
[location, [[referent, [[arg, [[root,'masa'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan, [[definite,positive]]]]]]]]]]

Example 5:

The following example is given to show how an attributive sentence can be represented in TMR. Note that, the main event of the TMR is an instantiated concept whose parent is an entity.

"Şu siyah, spor araba Ali'nin"
"That black, sport car is Ali's"

table-of-contents

speech-act	$speech\text{-}act_1$
heads	CAR_1
temp-rels	$temp\text{-}rel_1$

CAR_1

type	*sport*
color	*black*
owned-by	$HUMAN_1$
reference	*definite*
distance	*middle*
polarity	*positive*
aspect	$aspect_1$
time	$time_1$

$HUMAN_1$

type	*proper*
name	*Ali*

$aspect_1$

phase	*continue*
duration	*prolonged*
iteration	*single*
telicity	*true*

$time_1$

absolute	*present*

$speech\text{-}act_1$

type	*declarative*
scope	CAR_1
time	$time_2$

$time_2$

absolute	*past*

$temp\text{-}rel_1$

type	*extend*
arg_1	$time_1$
arg_2	$time_2$

F-Structure =

[[s-form,finite], [clause-type,attributive], [speech-act,declarative], [voice,active],
[verb, [[sense,positive], [mode,past], [root,'Ali'], [category,noun]]],
[arguments,
[[subject, [[referent, [[arg, [[root,'araba'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan, [[definite,positive]]],
[demons,şu]]],
[classifier, [[referent, [[arg, [[root,'spor'], [category,adjective]]]]]],
[modifier, [[qualitive, [[p-name,'siyah']]]]]]]]]

Example 6:

Next example is given to show how the *set* frame is utilized to represent a group of human that includes the speaker (denoted as 'we' in English').

"Yarın basketbol oynayacağız"

"We are going to play basketball tomorrow"

table-of-contents

speech-act	$speech\text{-}act_1$
heads	$PLAY_1$
temp-rels	$temp\text{-}rel_1$

$PLAY_1$		$time_1$	
agent	set_1	*day*	*tomorrow*
type	*basketball*		
polarity	*positive*	$speech\text{-}act_1$	
aspect	$aspect_1$	*type*	*declarative*
time	$time_1$	*scope*	$PLAY_1$
		time	$time_2$
set_1			
member-type	*HUMAN*	$time_2$	
cardinality	> 1	*absolute*	*past*
includes	*speaker*		
		$temp\text{-}rel_1$	
$aspect_1$		*type*	*after*
phase	*perfect*	arg_1	$time_1$
duration	*prolonged*	arg_2	$time_2$
iteration	*single*		
telicity	*true*		

F-Structure =
[[s-form,finite], [clause-type,declarative], [speech-act,declarative], [voice,active],
[verb, [[sense,positive], [mode,future], [root,'basketbol oyna'], [category,noun]]],
[arguments,
[[subject, [[referent, [[agr, [[person,first], [number,plural]]]]]]]]],
[adjuncts,
[[time, [[referent, [[arg, [[root,'yarın'], [category,adverb]]]]]]]]]]

Example 7:

The example below is given to demonstrate how more that one event in a TMR is related through thematic roles. Note that, in the following sentence the event $READ_1$ describes the *theme* of the main event $WANT_1$.

"Adam kitap okumak istedi"
"The man wanted to read a book"

table-of-contents	
speech-act	$speech\text{-}act_1$
heads	$WANT_1$, $READ_1$
temp-rels	$temp\text{-}rel_1$, $temp\text{-}rel_2$
coreferences	$coreference_1$

$WANT_1$		$aspect_2$	
experiencer	$HUMAN_1$	*phase*	*begin*
theme	$READ_1$	*duration*	*prolonged*
polarity	*positive*	*iteration*	*single*
aspect	$aspect_1$	*telicity*	*false*
time	$time_1$		

$aspect_1$
- *phase* — *perfect*
- *duration* — *momentary*
- *iteration* — *single*
- *telicity* — *true*

$time_1$
- *absolute* — *past*

$READ_1$
- *agent* — $HUMAN_2$
- *theme* — $BOOK_1$
- *polarity* — *positive*
- *aspect* — $aspect_2$
- *time* — $time_2$

$BOOK_1$
- *reference* — *indefinite*

$coreference_1$
- *speaker*, $HUMAN_1$, $HUMAN_2$

$time_2$
- *absolute* — *past*

$speech\text{-}act_1$
- *type* — *declarative*
- *scope* — $WANT_1$
- *time* — $time_3$

$time_3$
- *absolute* — *past*

$temp\text{-}rel_1$
- *type* — *after*
- arg_1 — $time_2$
- arg_2 — $time_1$

$temp\text{-}rel_2$
- *type* — *after*
- arg_1 — $time_3$
- arg_2 — $time_2$

F-Structure =

```
[[s-form,finite], [clause-type,predicative], [speech-act,declarative], [voice,active],
 [verb, [[sense,positive], [mode,past], [root,‘iste’], [category,verb]]],
 [arguments,
     [[subject, [[referent, [[arg, [[root,‘adam’], [category,noun]]],
                            [agr, [[person,first], [number,plural]]]]]]],
      [dir-object,
          [[role,
               [[role,act],
                [arg, [[s-form,infinite], [clause-type,predicative],
                       [voice,active], [speech-act, declarative],
                       [verb, [[sense,positive], [mode,past],
                               [root,‘oku’], [category,verb]]],
                       [arguments,
                            [dir-object,
                                 [[referent, [[arg, [[root,‘kitap’], [category,noun]]],
                                              [agr, [[person,third],
                                                     [number,singular]]]]]]]]]]]]]]]]
```

Example 8:

The following example is given to demonstrate how more that one events are related in a TMR through contextual boundedness. In the example below, the

event $READ_1$ provides extra information about the *theme* of the main event $REQUIRE_1$.

"Adam o çocuğun okudugu kitabı istedi"

"The man required the book that child was reading"

table-of-contents

speech-act	$speech\text{-}act_1$
heads	$REQUIRE_1$, $READ_1$
temp-rels	$temp\text{-}rel_1$, $temp\text{-}rel_2$
coreferences	$coreference_1$

$REQUIRE_1$

agent	$HUMAN_1$
theme	$BOOK_1$
polarity	*positive*
aspect	$aspect_1$
time	$time_1$

$HUMAN_1$

type	*common*
gender	*male*
age	≥ 18
reference	*definite*

$aspect_1$

phase	*perfect*
duration	*momentary*
iteration	*single*
telicity	*false*

$time_1$

absolute	*past*

$READ_1$

agent	$HUMAN_2$
source	$BOOK_2$
polarity	*positive*
aspect	$aspect_2$
time	$time_2$

$HUMAN_2$

type	*common*
gender	*unknown*
age	≤ 12
reference	*definite*
distance	*far*

$BOOK_1$

reference	*definite*

$aspect_2$

phase	*perfect*
duration	*prolonged*
iteration	*single*
telicity	*true*

$time_2$

absolute	*past*

$speech\text{-}act_1$

type	*declarative*
scope	$REQUIRE_1$
time	$time_3$

$time_3$

absolute	*past*

$temp\text{-}rel_1$

type	*after*
arg_1	$time_1$
arg_2	$time_2$

$temp\text{-}rel_2$

type	*after*
arg_1	$time_3$
arg_2	$time_1$

$coreference_1$

$BOOK_1$, $BOOK_2$

```
F-Structure =
  [[s-form,finite], [clause-type,predicative], [speech-act,declarative], [voice,active],
   [verb, [[sense,positive], [mode,past], [root,'iste'], [category,verb]]],
   [arguments,
       [[subject, [[referent, [[arg, [[root,'adam'], [category,noun]]],
                               [agr, [[person,third], [number,singular]]]]],
                  [specifier, [[quan, [[definite,positive]]]]]]],
        [dir-object,
            [[role,
                 [[role,theme],
                  [arg, [[s-form,participle], [clause-type,predicative],
                         [voice,active], [speech-act, declarative],
                         [verb, [[sense,positive], [mode,past],
                                 [root,'oku'], [category,verb]]],
                         [arguments,
                             [[subject,
                                  [[referent, [[arg, [[root,'çocuk'], [category,noun]]],
                                               [agr, [[person,third], [number,singular]]]]],
                                   [specifier, [[quan, [[definite,positive]]],
                                                [demons,o]]]]],
                              [dir-object,
                                  [[referent, [[arg, [[root,'kitap'], [category,noun]]],
                                               [agr, [[person,third],
                                                      [number,singular]]]]]]]]]]]]]]]]]]]
```

Example 9:

The next example is given to demonstrate how domain relations can be used to relate events in a TMR and the complex sentence representation constructed for the sentence below.

"Ali çalışmadığı için fizik dersinden kaldı"
"Since Ali did not study, he failed his physics course"

table-of-contents

speech-act	$speech\text{-}act_1$
heads	$STUDY_1$, $FAIL_1$
temp-rels	$temp\text{-}rel_1$, $temp\text{-}rel_2$
domain-rels	$domain\text{-}rel_1$
coreferences	$coreference_1$

$STUDY_1$

agent	$HUMAN_1$
polarity	*negative*
aspect	$aspect_1$
time	$time_1$

$aspect_2$

phase	*perfect*
duration	*prolonged*
iteration	*single*
telicity	*false*

$HUMAN_1$

type	*proper*
name	*Ali*

$aspect_1$

phase	*perfect*
duration	*prolonged*
iteration	*multiple*
telicity	*true*

$time_1$

absolute	*past*

$FAIL_1$

agent	$HUMAN_2$
patient	$COURSE_1$
polarity	*positive*
aspect	$aspect_2$
time	$time_2$

$COURSE_1$

type	*physics*
owned-by	$HUMAN_3$
reference	*definite*

$coreference_1$

$HUMAN_1$, $HUMAN_2$, $HUMAN_3$

$time_2$

absolute	*past*

$speech\text{-}act_1$

type	*declarative*
scope	$FAIL_1$
time	$time_3$

$time_3$

absolute	*past*

$temp\text{-}rel_1$

type	*after*
arg_1	$time_2$
arg_2	$time_1$

$temp\text{-}rel_2$

type	*continue*
arg_1	$time_2$
arg_2	$time_3$

$domain\text{-}rel_1$

type	*causal*
arg_1	$STUDY_1$
arg_2	$FAIL_1$

F-Structure =

```
[[type,linked],
 [linked-relation,icin],
 [arg1,
    [[s-form,finite], [clause-type,predicative],
     [voice,active], [speech-act,declarative],
     [verb, [[sense,negative], [tense,past], [root,'çalıs'], [category,verb]]],
     [argumnets,
        [[subject, [[referent, [[arg, [[root,'Ali'], [category,noun]]],
                               [agr, [[person,third], [number,singular]]]]]]]]]],
 [arg2,
    [[s-form,finite], [clause-type,predicative],
     [voice,active], [speech-act,declarative],
     [verb, [[sense,positive], [tense,past], [root,'kal'], [category,verb]]],
     [argumnets,
        [[subject, [[referent, [[agr, [[person,third], [number,singular]]]]]],
         [dir-object, [[referent, [[arg, [[root,'ders'], [category,noun]]],
                                  [agr, [[person,third], [number,singular]]]]],
                      [classifier, [[referent, [[agr, [[root,'fizik'],
                                                       [category,noun]]]]]],
                      [specifier, [[quan, [[definite,positive]]]]]]]]]]]]]
```

Example 10:

The following example is given to demonstrate how adverbial clauses can be represented in a TMR. Note that, the feature *value* of a temporal relation frame is introduced to represented the consecutive occurrences of the events $READ_1$ and GO_1.

"Ali notu okur okumaz okula gitti"
"Ali went to the school as soon as he read the note"

table-of-contents

speech-act	$speech\text{-}act_1$
heads	GO_1, $READ_1$
temp-rels	$temp\text{-}rel_1$, $temp\text{-}rel_2$
coreferences	$coreference_1$

GO_1

agent	$HUMAN_1$
goal	$LOCATION_1$
polarity	*positive*
aspect	$aspect_1$
time	$time_1$

$HUMAN_1$

type	*proper*
name	*Ali*

$LOCATION_1$

type	*school*
reference	*definite*

$aspect_1$

phase	*begin*
duration	*prolonged*
iteration	*single*
telicity	*true*

$time_1$

absolute	*past*

$READ_1$

agent	$HUMAN_2$
source	$NOTE_1$
polarity	*positive*
aspect	$aspect_2$
time	$time_2$

$NOTE_1$

reference	*definite*

$aspect_2$

phase	*perfect*
duration	*prolonged*
iteration	*single*
telicity	*true*

$time_2$

absolute	*past*

$speech\text{-}act_1$

type	*declarative*
scope	GO_1
time	$time_3$

$time_3$

absolute	*past*

$temp\text{-}rel_1$

type	*after*
arg_1	$time_1$
arg_2	$time_2$
value	*leq*0.1

$temp\text{-}rel_2$

type	*after*
arg_1	$time_3$
arg_2	$time_1$

$coreference_1$

$HUMAN_1$, $HUMAN_2$

```
F-Structure =
    [[s-form,finite], [clause-type,predicative], [speech-act,declarative], [voice,active],
     [verb, [[sense,positive], [mode,past], [root,'git'], [category,verb]]],
     [arguments,
         [[subject, [[referent, [[arg, [[root,'Ali'], [category,noun]]],
                                    [agr, [[person,third], [number,singular]]]]]]],
          [goal, [[referent, [[arg, [[root, 'okul'], [category,noun]]],
                                 [agr, [[number,singular], [person,third]]]]],
                    [specifier, [[quan, [[definite,positive]]]]]]]],
     [adjuncts,
         [[time,
              [[adv-type,as-soon-as],
               [argument,
                   [[s-form,adverbial], [clause-type,predicative],
                    [voice,active], [speech-act,declarative],
                    [verb, [[sense,positive], [tense,past], [root,'oku'], [category,verb]]],
                    [argument,
                        [[dir-object, [[referent, [[arg, [[root,'not'], [category,noun]]],
                                                      [agr, [[person,third],
                                                             [number,singular]]]]],
                                          [specifier, [[quan, [[definite,positive]]]]]]]]]]]]]]]]
```

Example 11:

This example is given to show how a yes-no question is represented in a TMR. Note that, the type of the *speech-act* frame is chaned to *interrogative*.

"Çocuk okula gitti mi?"

"Did the child go to the school"

table-of-contents	
speech-act	$speech\text{-}act_1$
heads	GO_1
temp-rels	$temp\text{-}rel_1$

GO_1	
agent	$HUMAN_1$
goal	$LOCATION_1$
polarity	*positive*
aspect	$aspect_1$
time	$time_1$

$HUMAN_1$	
type	*common*
gender	*unknown*
age	≤ 12
reference	*definite*

$aspect_1$	
phase	*perfect*
duration	*prolonged*
iteration	*single*
telicity	*false*

$speech\text{-}act_1$	
type	*interrogative*
scope	GO_1
time	$time_2$

$time_2$	
absolute	*past*

$LOCATION_1$
- *type* — *school*
- *reference* — *definite*

$time_1$
- *absolute* — *past*

$temp\text{-}rel_1$
- *type* — *after*
- arg_1 — $time_2$
- arg_2 — $time_1$

F-Structure =

[[s-form,finite], [clause-type,predicative], [speech-act,interrogative], [voice,active],
[verb, [[sense,positive], [mode,past], [root,'git'], [category,verb]]],
[question, [[type,yes-no]]],
[arguments,
[[subject, [[referent, [[arg, [[root,'çocuk'], [category,noun]]],
[agr, [[person,third], [number,singular]]]]],
[specifier, [[quan, [[definite,positive]]]]]]],
[goal, [[referent, [[arg, [[root, 'okul'], [category,noun]]],
[agr, [[number,singular], [person,third]]]]],
[specifier, [[quan, [definite,positive]]]]]]]]]

Example 12:

Following example is given to show how a wh-question type sentence is represented in a TMR. Note that, the question implies *agent* to be unknown and *focus* frame is used denote the argument which is the topic of the question.

"Camları kim kırdı"
"Who broke the windows"

table-of-contents
- *speech-act* — $speech\text{-}act_1$
- *heads* — $BREAK_1$
- *temp-rels* — $temp\text{-}rel_1$

$BREAK_1$
- *agent* — *unknown*
- *patient* — set_1
- *polarity* — *positive*
- *aspect* — $aspect_1$
- *time* — $time_1$

set_1
- *member-type* — $WINDOW_1$
- *cardinality* — > 1

$time_1$
- *absolute* — *past*

$speech\text{-}act_1$
- *type* — *interrogative*
- *scope* — $BREAK_1$
- *time* — $time_2$
- *focus* — $focus_1$

$time_2$
- *absolute* — *past*

$WINDOW_1$		$focus_1$	
reference	*definite*	*scope*	$BREAK_1.agent$
		value	1
$aspect_1$			
phase	*perfect*	$temp\text{-}rel_1$	
duration	*prolonged*	*type*	*after*
iteration	*multiple*	arg_1	$time_2$
telicity	*true*	arg_2	$time_1$

F-Structure =

[[s-form,finite], [clause-type,predicative], [speech-act,interrogative], [voice,active],
[verb, [[sense,positive], [mode,past], [root,'kır'], [category,verb]]],
[question, [[type,wh], [const,agent]]],
[arguments,
[[goal, [[referent, [[arg, [[root, 'cam'], [category,noun]]],
[agr, [[number,plural], [person,third]]]]],
[specifier, [[quan, [[definite,positive]]]]]]]]]