1991

ETRANS: A English-Thai translator

Nuntaporn Warote

Rochester Institute of Technology
Department of Computer Science

ETRANS:
An English-Thai Translator

by

Nuntaporn Warote

A thesis, submitted to
The Faculty of the Department of Computer Science,
in partial fulfillment of the requirements for the degree of
Master of Science in Computer Science

Approved by: Professor John A. Biles

Professor Kevin Donaghy

Professor Peter G. Anderson

May 1991

Title of Thesis: ETRANS: AN ENGLISH-THAI TRANSLATOR

I NUNTAPORN WARUTE prefer to be contacted each time a request for reproduction is made. I can be reached at the following address:

122/7 Soi Rewadee
Rama VI Road
Phayathai, Bangkok 10400
Thailand

Date: May 28, 1991

ABSTRACT

ETRANS is an experimental English-Thai machine translation (MT) system that translates a simple English sentence into a grammatically correct Thai sentence. The entire system is written in C-Prolog, and runs on UNIX systems. The MT strategy taken by ETRANS is an interlingual strategy with a parser for English and a generator for Thai. The parser creates a semantic representation equivalent to the meaning of the English sentence. A generator then interprets the semantic representation into Thai. ETRANS employs frames as a means for representing knowledge, and an augmented transition network (ATN) as the linguistic framework for analyzing and generating sentences.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

The work reported in this thesis is an attempt to implement a computer program capable of performing the task of translation. In particular, the system described herein, ETRANS, is an experimental machine translation (MT) system between English and Thai. The MT strategy taken by ETRANS is an interlingual MT strategy with two translation stages: first, an English sentence is analyzed and mapped onto a semantic representation indicating the meaning of the input sentence, and then the semantic representation is translated back into Thai sentence. In general terms, the work is concerned with the *analysis* of English sentences, the construction of a *semantic representation*, and the *generation* from the representations into Thai sentences.

The analyzer or parser operates by interleaving syntactic and semantic processing. It uses an augmented transition network (ATN) to perform a syntactic analysis (Woods, 1970), with calls to semantic procedures at various points to construct, using semantic information from the lexicon, one or more semantic structures for the constituent just recognized. If no coherent structures can be produced, the parser backtracks, thus avoiding following an incorrect path.

The semantic procedures are responsible for the resolution of word-sense, prepositional phrase (PP) attachment, and referential ambiguities.

Word-sense selection is based on semantic markers, selectional restrictions and verb case structures. The resolution of PP attachment is based on the case preferences of verbs, nouns, and prepositions. The referential disambiguation is accomplished by examining previous sentences.

The semantic representation produced by the parser is a structure providing a meaning of the input text with case-labeled components centered on the main verb element, each characterized in terms of primitive concepts and expressing both the meaning of a constituent and its function in the overall textual unit. Primitive concepts in this system are word-sense meanings. Case labels or semantic relations are those that the verb of a sentence has with its subject, objects, and PP arguments.

The generator is thus designed to interpret the semantic representation back into natural language, and specifically into Thai. The generating process is accomplished by another ATN. It treats the semantic representation as a transition network, traverses it, and evaluates each part of the semantic structure it encounters. The result would be either a Thai string, which would be added to the text being constructed, or a failure, which would cause the generator to back up and try an alternate path.

The representation of knowledge used by ETRANS is based on the notion of frames (Minsky, 1975). The frame representation language is the *Enhanced Frame Package* (Bhandari, 1989), which was developed at the Rochester Institute of Technology. It implements frames as an abstract data type and can be incorporated into C-Prolog programs.

The organization of the thesis is as follows: Chapter 2 presents the theoretical background on which ETRANS is based. Chapter 3 covers the

design and implementation of ETRANS. Chapter 4 discusses the evaluation method of the translated results and an evaluation of those results. Chapter 5 includes a summary of what has been achieved, directions for future work and a brief discussion of the future of MT.

CHAPTER 2

BACKGROUND

This chapter presents the theoretical issues on which the design of ETRANS is based. These issues are the choice of machine translation strategy, representation issues, and parsing and generating strategies.

2.1 Machine Translation Strategy

Today the major strategic decision that helps to guide research and development in Machine Translation (MT) is the choice between the transfer strategy and the interlingual strategy (Nirenburg, 1987). Under the transfer strategy a sentence in the source language (SL) is mapped into an internal representation, after which the transfer is made at both the lexical and structural levels into corresponding structures in the target language (TL), and then the translation is generated. The internal representation varies from purely syntactic deep structure markers to syntactic-semantic (such as case frame information) structures. The transfer module is developed for a specific source and target language pair. This entails relative inefficiency in a multilingual environment, since a transfer module has to be written for every such pair. Moreover, the transfer rules for mapping a SL internal representation into a TL internal representation are complex when the two languages are totally different.

An alternative to the transfer strategy, and the one taken by

ETRANS, is to make the translation with the help of a universal or language-independent representation called *interlingual.* Under the interlingual strategy the translation is in two stages: in the first stage the SL text is fully analyzed into an interlingual representation, and in the second stage the same interlingual is used to generate the TL text.

The interlingual MT systems fall into two classes: the syntactic approaches and those inspired by Artificial Intelligence (AI). The former was taken by the Russian-French project by CETA in Grenoble and the German-English project of the Linguistics Research Center (LRC) at the University of Texas (Hutchins, 1986). The basic stages of translation were: analysis of SL texts into an intermediary representation, and production of TL texts from the intermediary representation. The intermediary representation was declared universal (interlingual) and was restricted to syntactic structures. There was no attempt to decompose lexical items into semantic primitives. Consequently, conversion of vocabulary from SL into TL was made through a bilingual dictionary operating essentially at the lexical level. The latter is influenced by the basic AI argument that at some stage, translation involves the understanding of a SL text in order to convey its meaning in a TL text. This means that texts must be given semantic or conceptual representations (independent of any language), that parsing must be based on semantic criteria, and that the knowledge databases must be used to assist in the interpretation of texts.

The ETRANS translation strategy is AI-based interlingual by definition. There is a separate parser and generator for each source and target language. The parser is designed to interface with a concurrent semantic

analysis. The semantic analysis is responsible for building an interlingual representation (called in this thesis, semantic representation, SR) which is used to guide the operation of the parser. Its interlingual is designed to be independent of any language; however, it also carries syntactic information so the syntactic style of the original text is preserved in the translation.

2.2 Representation

The general task of any MT system is to convert a surface sentence in one language into some kind of internal representation of the meaning of the sentence, and then to translate back out into another language. In order to do this, an MT system must have some formal way to represent the meaning of input text so the output text can be generated. This section describes the basis for representing meaning.

2.2.1 Frames

Meaning in this thesis is represented by frames (Minsky, 1985). The frame representation language used is the *Enhanced Frame Package* (EFP) (Bhandari, 1989), a modified version of the *Frame Package* (Hiss, 1987) developed at the Rochester Institute of Technology. The Enhanced Frame Package can be incorporated into C-Prolog programs, and it consists of a knowledge base of frame clauses, together with operators *add_value_to_slot*, *remove_value_from_slot*, and *return_value_from_slot* to add, delete, and retrieve information to/from frames.

A frame contains a name, and a set of *slots*. A frame's important substructures and its relation to other frames is defined in its slots. A slot has a *slot-name*, and a set of *facets*. The values for a slot are determined by a

set of facets. A facet is composed of a *filler* or *value,* cardinality, a description containing what sort of values it allows, a default value, and a set of attached procedures. Values in a certain slot can take other frames as values. Figure 2-1 shows the EFP syntax.

framename:[slot1name:[facet1name:facetvalue
facet2name:facetvalue,
.
.
facetNname:facetvalue],
slot2name:[list of facets]
.
.
slotNname:[list of facets]]

Figure 2-1. The Enhanced Frame Package syntax

Frames are related to one another in a *hierarchy.* A frame that is connected to another frame one level down in the hierarchy is called the *parent* of that frame, and the lower frame is the *child.* Relationships between frames are established by two special slots: *is_a* slot and *in_of* (instance of) slot. An is_a slot indicates a class-subclass link; it connects the parent and children, where the parent represents a general group of objects called a *class* and the children represent different types, or *subsets,* of the parent objects. The in_of slot indicates the member link; it connects parents and children, where the child is the same type of object as the parent, but is a specific instance of the parent. Instances or tokens are created during the execution of the program. These two slots provide the inheritance capabilities. Attributes associated with frames that represent classes of objects are inherited by the subclasses and members of these frames.

2.2.2 Case Structures

Meaning representation in ETRANS is based on a case system (Fillmore, 1968). The notion of cases is based on the view of a clause as the description of an event, which is specified by the verb, and whose participants are described by the noun phrases of the clause; the relationship of each participant to the event is called the *case* of the participant.

Representation of meanings as case-frames is a feature of many language-processing systems. Opinions differ, however, on the exact level of depth desirable for a system to seek to attain. Three case systems that influenced the design of that used in ETRANS are: Fillmore's case frame structures, Schank's conceptual dependency theory, and Wilks' preference semantics.

Fillmore (Fillmore, 1968) proposes that verbs be classified according to *sentence types* or *case frames.* A case frame tells what case relationship may exist between a verb and its noun, and for each case there are three possibilities: it is required with this particular verb, it is optional, or it is not allowed.

Schank's conceptual dependency (CD) theory (Schank, 1973) works in terms of *conceptualizations,* which do not involve any of the surface words of the language. In CD, a small number of concepts corresponding to primitive acts can be used to construct meaning representations for most descriptions of events. These primitive concepts are simple actions of the kind "move a body part" (MOVE), "transfer a physical object" (PTRANS), "produce a sound" (SPEAK), and "transfer mental information" (MTRANS). The primitive acts together with the required conceptual cases: *actor, object,*

direction, and either *recipient* or *instrument* are the components of meaning representation with a unique representation feature.

Wilks' theory of preference semantics (Wilks, 1975) is based on semantic primitives that Wilks called *elements.* These elements are 70 primitive semantic units used to express the semantic entities, states, qualities, and actions about which humans speak and write. These elements are used to build up *formulas* - each formula expresses the sense of an English word and is composed of elements combined into a binarily bracketed list-structure. A typical definition using the primitives is the following definition for one sense of the word *drink* (Wilks, 1979):

```
(DRINK:    ( (*ANI SUBJ)
             ((FLOW STUFF) OBJ)
             ((SELF IN) (((*ANI (THRU PART)) TO)
             (BE CAUSE) )
```

This says roughly that DRINKing is a CAUSing to MOVE, preferably done by an ANImate SUBJect (agent) and to a liquid (FLOW STUFF), TO a particular ANImate aperture (THRU PART), and INto the SELF (the animate agent). The case primitives: SUBJ, OBJ, and TO are the name of relations in the semantic representation.

The text structures in the system are semantic templates (together with semantic ties between them). A template is a network of such word-sense formulas, containing at least an agent, action and object formula. Thus the template for *John drinks beer* would be structured as follows:

[john] <------> [drink] <------> [beer]

Here the bracketed English words should be imagined as being replaced by the semantic formulas representing their appropriate sense.

Relations among templates are indicated at a higher level of structure.

2.3 Parsing and Generating Mechanisms

To represent natural language structure, ETRANS uses the model known as the augmented transition network (ATN). The ATN was originally proposed by Woods (Woods, 1970) as a device for the analysis of natural language. It is a development of the basic transition network grammar: a finite state transition diagram with named states connected by arcs, the arcs themselves labeled with terminal symbols (words), or state names, i.e. a recursion (non-terminal symbols). A successful parse is associated with a complete path through the network, starting at the initial state, and terminating at one of the final states of the grammar. The augmentation comes from adding to the arcs an open-ended set of structure-building and register-setting actions as well as test conditions. Thus registers can be assigned arbitrary structures or partial analyses; they can at any later stage of the analysis be interrogated, thereby determining the further course of the analysis; and they can also be modified.

On this basis, ETRANS uses the ATN framework as the parsing mechanism. There are, however, differences between the ATN implemented in this thesis and Woods' original ATN. The new implementation of ATN incorporates syntactic and semantic processing, and produces the semantic representation indicating the meaning of the sentence. The semantic procedures are added as actions to be performed upon the completion of noun phrases, verb groups, prepositional phrases, and clauses. There are no structure-building actions for building partial structures on the arcs of the

grammar itself. The syntactic constituents just parsed are passed as arguments to the semantic procedures whose task is to build the semantic structures representing the meaning of a piece of text.

ETRANS also uses the ATN formalism for sentence generation. ATN for generation has essentially the same organization as its parsing counterpart, with the obvious exception that where a parser will scan a word from the input sentence as an action on an arc, the generator will produce one. A generation scheme taken by ETRANS is similar to that described by Shapiro, 1982. That system uses an ATN grammar whose input is a node of a semantic network and whose output is a linear string, a sentence describing the node. The grammar controls the syntax of the generated sentences, but bases specific decisions on the structural properties of the semantic network and the information contained therein (Shapiro, 1982).

In ETRANS, however, input to the ATN grammar is the frame representation whose slots are equivalent to nodes of the semantic network, and the value in each slot represents a concept to be expressed as part of natural language statement.

CHAPTER 3

IMPLEMENTATION

ETRANS is a MT system for translating English sentences into grammatical Thai sentences. The entire system is written in C-Prolog, and runs on an AT&T UNIX system. The MT approach taken by ETRANS is an interlingual approach, which contains a parser for English and a generator for Thai. The diagram in Figure 3-1 represents the overall structure of the translation system. The parser and the generator communicate with each other using the semantic representation indicating the meaning of the input sentence. The information needed by the two processes are supplied by the lexicon, which relates words of natural languages to their grammatical information and their underlying concepts. Since the lexicon contains a limited number of words, the system thus provides a preprocessing procedure to check words in the input sentence against words in the lexicon. ETRANS is only an experimental system. It works on an isolated sentence; multiple translation is, therefore, accounted for.

3.1 The Lexicon

The lexicon component of ETRANS consists of three types of lexicon: the concept lexicon and the analysis and generation lexicons of natural languages.

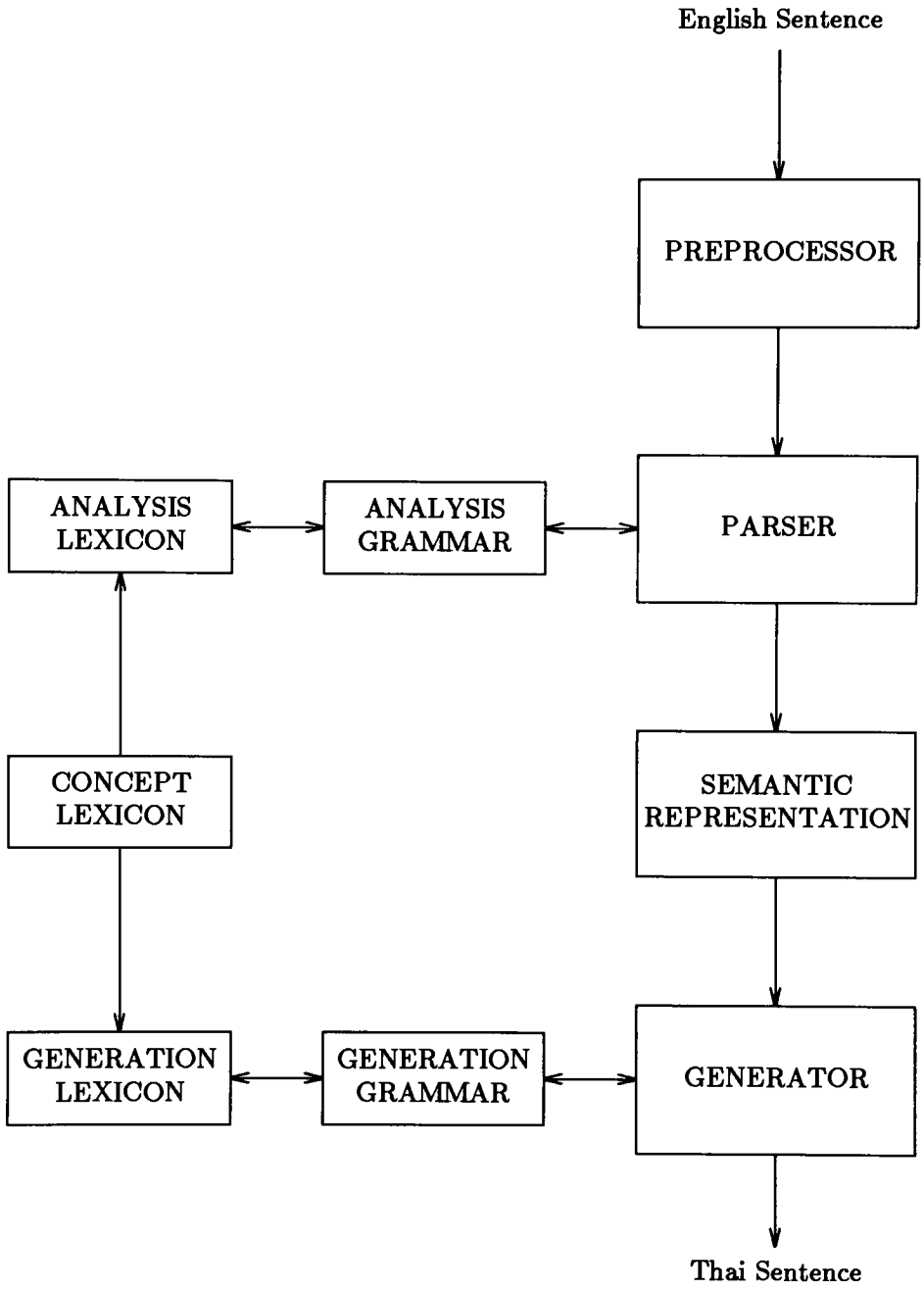

Figure 3-1. Translation Design in ETRANS

3.1.1 The Concept Lexicon

The concept lexicon determines the inventory of semantic features or primitive concept types used in the entries of the analysis and generation lexicons. It forms the tangled inheritance hierarchy or isa hierarchy shown in Figure 3-2. Each entry is represented in the form of concepts and roles. A concept is defined by its attributes, which consist of two parts: slots and value restrictions. The value restriction is a concept that defines the range of possible fillers for the attribute. The slot defines the relation of the filler to the concept being defined. A path of concepts from the root to the action node is presented below:

```
frame($thing:[]).
```

This is the root of the *isa* hierarchy. The prefix "$" indicates that *thing* is a primitive concept type in the concept lexicon as opposed to *thing* which is a natural language word whose grammatical category is a noun.

```
frame($event:[
              is_a:[value:[$thing]],
              $object:[type:$thing]],
              voice:[default:[active]],
              tense:[default:[pres]],
              aspect:[default:[nil]],
              negation:[default:[n]],
              modality:[default:[nil]]]).
```

At this level the *is_a* slot indicates the pointer to the node's parent in the hierarchy. The *event* frame represents the actions and states. One of the properties common to all events is the *object* case, an obligatory case found with every verb, representing the thing being acted upon, or the thing

in the state described by the verb. Another is the modality information, that is, voice, tense, aspect, negation, and modalization. *Type:thing* in the *object* slot specifies that the slot filler is required to have the value which is an instance of the class *thing*. *Default:[...]* indicates that if information needed is not to be found in the value facet, then use the value in the default facet.

```
frame($action:[
            is_a:[value:[$event]],
            $agent:[type:$animate]]).
```

The *action* frame represents the meaning of all actions. It has slots that correspond to cases of a case grammar (agent, object, etc). The *agent* slot, representing the person or thing performing an action, prefers the fillers of type animate. The *object* slot, inherited from the *event* frame, prefers the fillers of type thing. The modality information is also inherited from the *event* frame.

```
frame($ingest:[
            is_a:[value:[$action]],
            $object:[type:$ingobj]]).
```

This frame declares that only ingestible objects (concept type ingobj) such as food, medicines, etc. can be fillers for the *object* slot; the *agent* slot, inherited from the *action* frame, takes only an animate entity; and the modality information is inherited from the *event* frame.

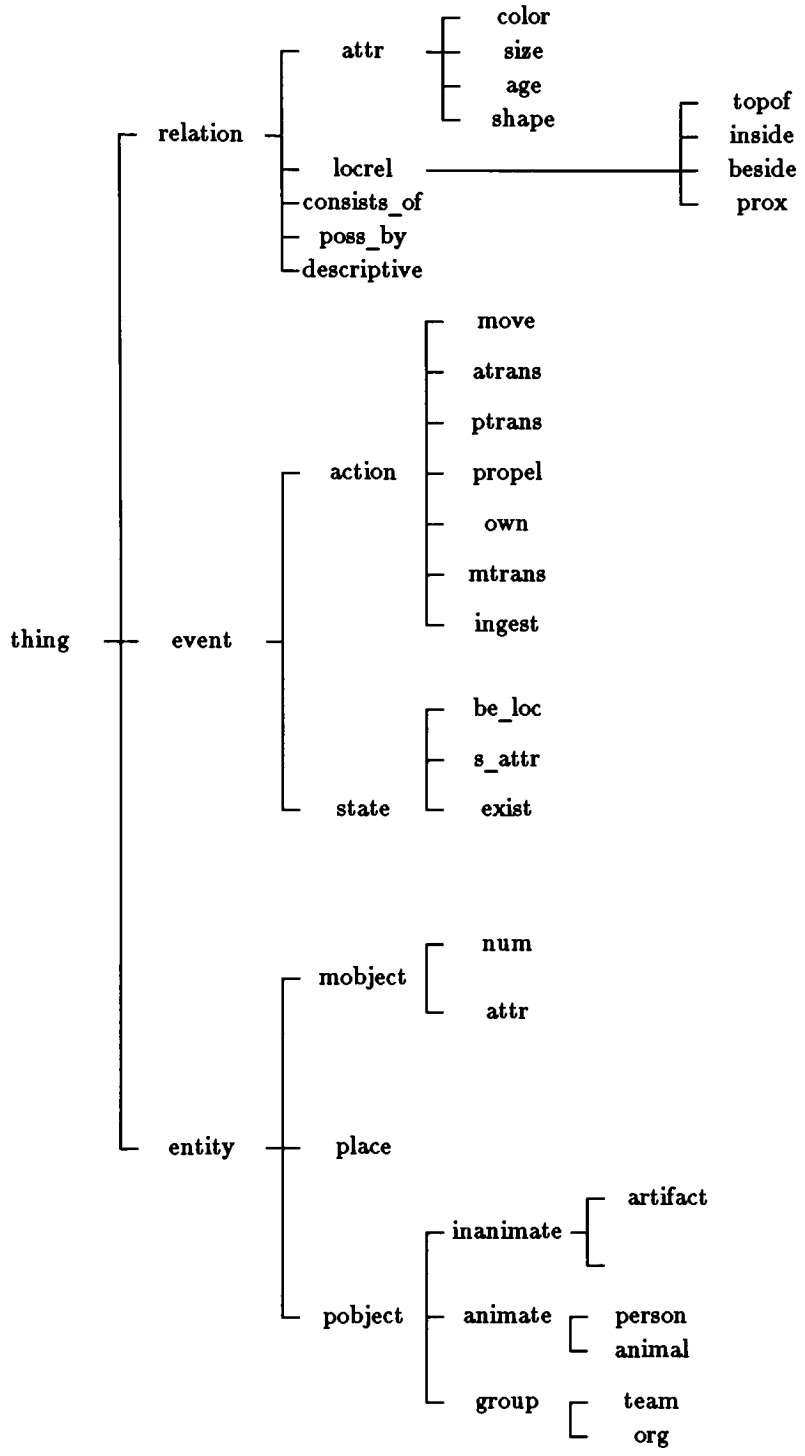

Figure 3-2. A Fragment of the *isa* Hierarchy.

3.1.2 The Analysis Lexicon

The analysis lexicon is indexed by English words that have associated with them both syntactic and semantic information. The syntactic information contains the part of speech or syntactic category that the word belongs to and a list of syntactic features of each part of speech the word can have. The descriptions of the features are presented in Figure 3-3. The semantic information contains a list of various senses of the word. Each word sense itself is a frame connected to the concept type frame in the concept lexicon. Words are represented in the system in the frame notation. Here is a template for an entry for a word in the analysis lexicon.

```
frame(word:[
              features:[value:[...]],
              wordsenses:[value:[sense1,sense2,...]].
```

The *features* slot gives the syntactic features of the word. The value contained in the features slot is a list of features. If a given word is an inflected form, its frame contains a slot *root* whose value is a pointer to the root word frame. Its syntactic features are obtained by activating the if_needed demon *root* in the *features* slot. *Root* takes additional two arguments: the list of features to be added, and the list of features to be removed. Call to this procedure causes the features of the root word to be changed. These changes are the syntactic features of the given word. Here are examples of the frame definitions of the words *go* and *went*.

```
frame(go:[
              features:[value:[verb,intrans,inf]],
              wordsenses:[value:[go1]].
```

```
frame(went:[
            root:[value:[go]],
            features:[if_needed:root([past],[inf])]]].
```

The demon root([past],[inf]) of the *went* frame indicates that the feature *inf* of the *go* frame is to be replaced by the feature *past*, but the rest is to remain the same.

Feature	Description
p1	first person
p2	second person
p3	third person
sin	singular number
plu	plural number
masc	masculine gender
fem	feminine gender
neut	neuter gender
obj	objective case
nom	nominative case
poss	possessive case
pronoun	pronominal noun
proper	proper noun
inf	infinitive tense
perf	perfect tense
prog	progressive tense
past	past tense
pres	present tense
fut	future tense
intrans	intransitive verb
trans	transitive verb
indobj	bitransitive verb
copula	copula verb

Figure 3-3. Description of Features for English.

Each root word frame has the *wordsenses* slot, whose values are various senses of the root word. Each word sense is a pointer to the word sense

frame representing a distinct meaning of the particular word. It is written as the word followed by a number to distinguish that sense from the other senses of the same word. For instance, the list of senses for hand is *hand1*, *hand2*; where *hand1* represents a noun referring to the body part (concept type *bodypart*) and *hand2* represents a verb referring to an act of giving by hand (concept type *atrans*).

```
frame(hand1:[
        is_a:[value:[$atrans]],
        surfaceword:[value:[hand]],
        features:[value:[verb,indobj]],
        thai:[value:[haixx1]]].

frame(hand2:[
        is_a:[value:[$bodypart]],
        surfaceword:[value:[hand]],
        features:[value:[noun]],
        thai:[value:[myu1]]].
```

Associated with each word sense frame is an *is_a* slot, whose value is the pointer to the concept type frame in the concept lexicon, a *surfaceword* slot, which specifies the natural language word for the particular word sense, a *features* slot, which is used in the process of disambiguation by the fact that different senses may be different syntactic categories, and a *thai* slot, which contains a list of equivalent Thai word senses and the conceptual case slots inherited from the corresponding type representations.

A word sense frame might be set up with its own slots instead of inheriting from its corresponding concept type. The constraints on these slots are often used to indicate the word sense's special requirements or implications, thereby causing that particular word sense to be selected rather than the other senses of the same surface word, or causing it to be discarded if its

constraints are not satisfied. For example, consider the frame representation for *take1* (and ingest sense of *take*):

```
frame(take1:[
            is_a:[value:[$ingest]],
            $object:[if_added:$drugnosourcedest],
            features:[value:[verb,trans]],
            surfaceword:[value:[take]],
            thai:[value:[kin1]]]).
```

In this representation, the concept that can fill the object slot for *take1* must satisfy a set of restrictions imposed by the *if_added* demon, *$drugnosourcedest*, or else *take1* will be rejected. The demon *$drugnosourceordest* is simply a Prolog procedure whose function is to check that the filler has the semantic feature *drug* and that a prepositional phrase whose preposition indicates the source or destination case does not exist.

3.1.3 The Generation Lexicon

The entries in the generation lexicon are indexed by Thai word-senses. Associated with each word sense is the pointer to the corresponding Thai word. The structure of each entry in this lexicon is the same as the one in the analysis lexicon. However, some entries may have a special slot *makeprep*, which is mainly used to introduce prepositions in the output string. Here is an example of the Thai word-sense frame *haixx* (give):

```
frame(haixx1:[
            is_a:[value:[$atrans]],
            makeprep:[value:[ [kaex,$recipient] ]],
            surfaceword:[value:[haixx]],
            features:[value:[verb,indobj]],
            english:[value:[give1]]]).
```

The *surfaceword* slot is a reference to a surface word frame; the pointer for *haixx1* is to the lexical entry *haixx*. The *features* slot contains a syntactic category and syntactic features. The *makeprep* slot indicates that it requires an insertion of a preposition *kaex* (for) to the output string if the semantic content of the case *recipient* is to be expressed. The *english* slot contains a list of equivalent English word senses.

3.2 Semantic Representation

The semantic representation (SR) is constructed by the parser and provides a meaning representation of the input sentence. It characterizes dependencies between an event (verb) and its arguments (noun and prepositional phrases).

The SR is represented in terms of frames in which the frame represents concepts and the slots represent relations between the frames they connect. Concepts are produced by obtaining tokens of the word-sense meanings in the analysis lexicon and then augmenting them by various attributes identified during the parsing process. Each token stands in the *in_of* relationship to its corresponding word-sense; it is shown to be a token by adding an underscore sign plus a number onto the end of a word sense, for instance *coach1_1* is a token for the noun referring to the person primitive type of *coach*, *coach1*. The attribute values must correspond to the data types listed as fillers for the corresponding slots either in the concept type frames or the word-sense frames. A typical frame for an event token is as follows. The *wordsense_id* in the *in_of* slot stands for the name of the corresponding word-sense.

```
frame(action_token_id:[
            in_of:[value:[wordsense_id]],
            $agent:[value:[object_token]],
            $object:[value:[object_token]]]).
```

As an example of the representation language consider the sentence *The coach bought three red balls.* The corresponding representation is shown in Figure 3-4.

```
frame(buy1_1:[
            in_of:[value:[buy1]],
            $agent:[value:[coach2_1]],
            $object:[value:[ball2_1]],
            voice:[value:[active]],
            tense:[value:[past]],
            negation:[value:[n]]]).

frame(coach2_1:[
            in_of:[value:[coach2]],
            number:[value:[sin]],
            ref:[value:[def]]]).

frame(ball2_1:[
            in_of:[value:[ball2]],
            $quantity:[value:[three1_1]],
            $color:[value:[red1_1]],
            number:[value:[sin]],
            ref:[value:[def]]]).

frame(three1_1:[in_of:[value:[three1]]]).

frame(red1_1:[in_of:[value:[red1]]]).
```

Figure 3-4. A Semantic Representation for *The coach bought three red balls.*

3.3 The Preprocessor

The preprocessor is the first module accessed when an input sentence has been entered for translation. There are three passes of lexical preprocessing. The first is a procedure *read_sent*, which produces a list of atoms built

from strings read from standard input. The second procedure, *split* allows a word to be replaced by series of words, for example *haven't* replaced by *have not.* The last is a dictionary look-up process and is known to the program as a procedure *check_word. Check_word* compares each input word against each lexical entry in the dictionary. If unknown or misspelled, the sentence must be reentered; there is no spelling correction or learning operation.

3.4 The Parser

The task of the parser is to convert a surface string encoding an English sentence into a well-formed meaning representation that can then be passed off to the generator. This section describes the syntactic and semantic processing of the parsing system.

3.4.1 The Control Structure

The overall control structure of the parser is that of an ATN parser with mixed top-down and bottom-up processing. The top-down processing starts with the syntactic recognition rules, which expect the next word in the input string of certain syntactic types. The rules for parsing a simple noun phrase, for example, state that if what comes before noun can be a determiner, then a determiner will be looked for even if the next word is a noun. In such a case it is said to have a failure. The parser then must go back to the previous state and try a different path. The bottom-up processing is determined by the semantic processing. It takes a syntactically well-formed constituent and tries to build portions of the final semantic structure, which are subsequently put together in a higher level semantic unit.

A Prolog program is responsible for maintaining the parsing process

during a strictly left-to-right sentential scan. It is written to incorporate semantics in such a way that the semantic structure is built during a single pass through the sentence. The emphasis is on the semantic processing with syntactic recognition taking place in the background and providing partial syntactic results for the semantic procedures. The semantic procedure builds a semantic structure immediately after a complete constituent, for instance a noun phrase or a clause, is recognized. This structure is stored in a variable until it can be given a conceptual role to play. If the parser sees the word is analyzable into more than one meaning structure, the parser will deliver them all to the next level of semantic computation, which will try to select the appropriate word meaning using syntactic, semantic, and local contextual cues.

3.4.2 Syntactic Processing

The syntactic processing is based on an ATN grammar whose task is to isolate the constituents of a clause, rather than to build a complete structure that gives the syntactic relationships between all the constituents. The constituents it recognizes are noun phrases, verb groups, prepositional phrases, and clauses.

The particular ATN used in ETRANS is diagramed in Figure 3-5. The sentences accepted by this ATN are simple declarative clauses. The clauses are simple in that they have no embedded clauses or any adverbial modifiers. The clause thus contains only verbs, nouns and prepositional phrases; the verbs may be active or passive and may have a perfective auxiliary; the noun phrases (including those in the prepositional phrases) may be

determined or non-determined or may be a pronoun; they may have a quantifier; and they may also have any number of adjectives.

3.4.3 Semantic Procedures and Semantic Processing

The semantic procedures operate within the boundaries of syntactically well-form constituent. Its function is to disambiguate surface lexical items, and to construct the semantic representation of the syntactic constituent at the current level of computation. The major semantic procedures are as follows:

build_np(...) is called as soon as the head noun is found. It contains the procedures that participate in the process of constructing the semantic representation of the noun phrase. The process starts by constructing an individual concept for each sense of the head noun. It then cycles through all the constituents in front of the head noun testing for compatibility between the available senses of those constituents and the head noun senses. The test is by inspecting the word-sense frames for those constituents for the preferred semantic class of the concept they modify.

build_clause(...) is the master semantic routine, which is organized around the verb and is called after the major clause components: subject, verb, and objects, are identified. It is handed the semantic contents of the clause components: *Subj, Verb,* and *Obj.* Normally, these are ambiguous concepts, and the procedure will have to select the sense of the verb, and the sense of the nouns; and the assignment of case roles to conceptual objects specified by the noun phrases of the sentence.

pp_attachment(...) is responsible for the analysis and structural dis-

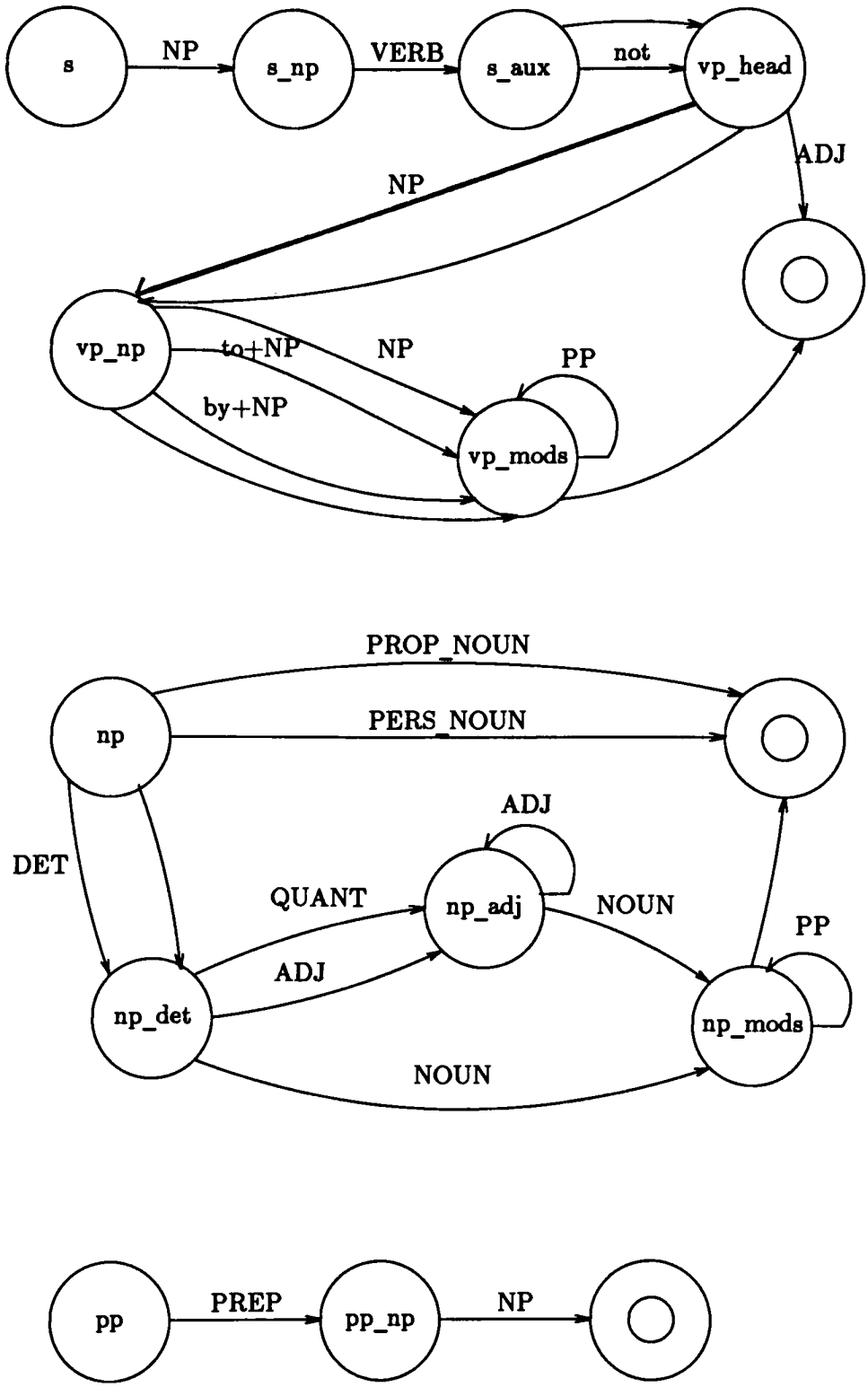

Figure 3-5. The ATN Used in ETRANS' Syntactic Analysis

tribution of the postnominal and postverbal prepositional phrases. Its goal is to attach the modifiers picked up by the syntactic parser either to a verb or to an immediately preceding noun phrase. This is done by determining lists of prepositional meanings attached to the noun, verb and preposition. More detail is described in Section 3.4.4.2

3.4.4 The Resolution of Ambiguity

The task of resolving the ambiguity falls wholly on the semantic procedures. Resolution happens as soon as subsequent words have provided enough information, and in any case it must occur by the end of the sentence. This section describes the word-sense, PP attachment, and referential ambiguities resolution components.

3.4.4.1 Word-sense Disambiguation

The resolution of word-sense ambiguity is handled by the two major independent semantic procedures: *build_np* and *build_clause*. They both check the semantic coherence and well-formness of a complete syntactic constituent, and deliver the corresponding meaning representation.

Normally, a noun phrase can not be disambiguated at the current level of computation. For example, *the green coach* where *green* can mean either the color green, *green1* or an inexperienced person, *green2*; and *coach* can mean either a passenger coach, *coach1* or a trainer, *coach2*. The phrase *the green coach* cannot be processed by *build_np* any further than cutting down the number of possible interpretations from four (2*2) to two. In such a case, a list of multiple meaning structures must be carried over to the next level of semantic computation.

For the disambiguation process to proceed, a special structure is constructed to hold all possible meaning structures so they can be explicitly accessible by other semantic procedures. This structure is indexed by a symbol written as the word *ambiguity* followed by a number. Here in the process of disambiguating the phrase *the green coach, build_np* constructs an ambiguous concept called *ambiguity_1* to hold all the tokens for the types of each sense of *coach*: *coach1_1* and *coach2_1*. Next, it applies the selectional restrictions in an attempt to disambiguate between *coach1_1* and *coach2_1*. This means the process examines the semantic definitions of *green* trying to match the meanings of green against *coach1_1* and *coach2_1*. Since *green1* is a characteristic of inanimate objects, to the class of which *coach1*, but not *coach2*, belong, an instance of *green1*, *green1_1*, is created and is then linked to the instance frame *coach1_1*. The match also found *green2* semantically compatible with *green2_1* so the instance frame *coach2_1* is also modified.

The same techniques underlie the design of the procedure *build_clause*, though the process is complicated by the fact that a word sense here contributes to the interpretation of the sentence as a whole, rather than of the word on its own. The function of *build_clause* is to disambiguate the sense of the verb as well as the sense of the nouns and to build the semantic representation. The selection (disambiguating) process first deletes the senses of the verb which are obviously incompatible with the current syntactic environment. For example, the word *run*, meaning to move on foot or to operate, can be disambiguated by seeing whether the verb of the sentence is tagged *intransitive* or *transitive*.

Further disambiguation is governed by selectional restrictions and

case structures. This is achieved by matching the semantic features that a verb expects for its subject and object against the features of the nouns which are in the sentence. For example, the verb *serve* in *The coach served the noodles*, whether *serve* referring to offering something for eating or drinking, or an act of serving in tennis. In this case *serve* can be easily disambiguated as to offer something by the fact that *the noodles* is a type of food, which is allowed to be filled in the *object* slot. The nearby disambiguating words may themselves be ambiguous; an example is *the coach*. The sense of *coach* as a passenger coach would be rejected in favor of the *serve* senses; they requires its *agent* to be a person.

Another important selectional restriction concerns oblique arguments. For example, the verb *take* has four senses, *take1* means swallow medicine (a concept type *ingest*); *take2* means to steal (a concept type *atrans*); *take3* means transport (a concept type *ptrans* which requires the appearance of either *source* or *destination* or both) and whose frame representation is shown below; and *take4* means swindle.

```
frame(take3:[
            is_a:[value:[$ptrans]],
            ...
            compulsory:[value:[
                [from,[$place],$destination,from1],
                [to,[$place,$group],$source,to1]]],
            features:[value:[verb,trans]],
            surfaceword:[value:[take]],
            thai:[value:[dooysarn1,aaw1,phaa1]]]).
```

The information on the oblique arguments, PP in this case, is encoded in a slot *compulsory*. A template [from,[$place],$destination,from1] represents a *from*-phrase, the *destination* of the verb if the prepositional

object is a *place*.

Consider a sentence *John took a ball to school*, where *take2* and *take3* are alternatives. Since there is a destination case (a PP, *to school*) given, *take3* is preferred, and so *john* and *ball* are easily resolved by way of usual slot constraint check.

3.4.4.2 Prepositional Phrase Disambiguation

The resolution of PP attachment ambiguity is handled by the procedure *pp_attachment*. The function tries to find the proper place for its attachment, whether to the verb or the preceding noun phrase. However, the syntactic processor prohibits verb phrase attachment of PP that occur between the subject and verb of a sentence, as in *The man in the red coat kicked the ball*, etc. PP in this position is always NP-attached.

The resolution of this ambiguity uses the case preferences (Wilks, 1985), encoded in a slot *preps*, of verbs, nouns, and prepositions. The case information for verbs and nouns are encoded into their word sense frames. Each case is a list of four elements. The first element is a list of prepositions that flag the case of preposition; the second is the preferred semantic class of the head noun of the prepositional phrase; the third is the case of preposition; and the fourth is the sense of the preposition. The word sense frame for one sense of the verb *buy* contains a recipient case, as in *John bought the dress for Mary*:

```
frame(buy1:[
            is_a:[value:[$atrans]],
            ...
            preps:[value:[
               [[for],[$person,$group],$recipient,for1]]]]).
```

The word sense frame for one sense of the noun *house* has a location case, as in *house by the school* (this case preference information is put into the system based on the assumption that people normally characterize immovable objects by their locations):

```
frame(house1:[
            is_a:[value:[$struc,$place]],
            preps:[value:[
               [by,[$org,$struc],$location,by3]]]]).
```

The case information for prepositions is encoded under the names of individual prepositions. Each case information frame is comprised of four elements. The first element represents the preferred semantic class of the head noun or the verb preceding the prepositional phrase to be attached; the second element is the case of the preposition; the third element is the preferred semantic class of the head noun of the prepositional phrase; and the fourth is the sense of the preposition. Below is the lexical entry for *by*:

```
frame(by:[
            features:[value:[prep]],
            preps:[value:[
               [$move,$conveyance,[$movable_inst],by1],
               [$move,$location,[$pobject],by2],
               ...
               [$action,$location,[$struc,$org],by3]]]).
```

The disambiguation process is in three stages. The first stage attempts PP attachment using the noun case preferences. The second stage attempts PP attachment using the verb case preferences. The third stage attempts PP attachment using the case preferences of the preposition, starting with the main sentence verb.

3.4.4.3 Referential Disambiguation

The resolution of this type of ambiguity is accomplished by the procedure *find_referents*. Currently, *find_referents* works on a pronoun that refers to the previous mentioned concept in the previous sentence. The way the procedure deal with pronouns is very simple. Only three pronouns *he*, *she*, and *it* are known to the system, and these are treated as though they are *the man*, *the woman*, and *the animal* or *the inanimate object* respectively.

As an example, *John gave banana to the monkey. It was ripe.* The procedure *find_referents* is called as soon as *it* is recognized to find all possible referents for *it*. *Find_referents* is handed an information on what kind of referent to look for, in this case *it* refers to the previous mentioned animal or inanimate object.

Find_referents then searches the previous sentence which is held in the frame *eventlist*, selects the concepts previously seen which match up the feature of the pronoun *it*. When a list of possible referents is returned, the pronoun's concept is replaced with the concept of the antecedent if there is only one possible referent, or with an ambiguous concept of the possible referents if there are more than one. For this example, an ambiguous concept is constructed to hold *banana* and *monkey*. Further disambiguation is accomplished by the procedure *build_clause* which will select *banana* as the correct referent for *it* because of the semantic requirements of *ripe*.

3.5 The Generator

Generation of Thai sentences from a semantic representation (SR) is viewed as the creation of a linear surface string that describes the case roles of the SR. The syntax of Thai sentence is controlled by the second ATN. The arcs on this ATN are different from the ones used in the parser since the generator is a transducer from the SR into a linear string, whereas the parser is a transducer from a linear string into the SR. Tests on the arcs of the ATN examine the case roles of the SR and fail or succeed accordingly. In this way, states of the ATN can decide what surface form is most appropriate for describing a case role, while different ATN states may generate different surface forms to describe the same case role. The action performed during the successful transition through the arc is simply the concatenation onto the surface string being built.

During the generation process, certain syntactic knowledge is made available to the program; such as normally the agent plays the role of the subject of the sentence, the object plays the role of the direct object of the sentence, the recipient is the indirect object, etc.

3.5.1 Thai Grammar

This section describes an outline of Thai syntax used in the sentences presented in this thesis. Thai is like English in that the typical sentence contains subject, verb and object in that order. But Thai attributive constructions differ from those of English in that the head must always precede the attribute. The passive construction is generated by adding the word *tuukx* (like be-en in English) before the main verb of the sentence.

Thai makes no use of inflections. Plurality if not indicated by the text of the sentence, may be indicated by adding numeral words. The Thai verb itself does not indicate tense. When the verb is used in various tenses, it must be accompanied by certain auxiliaries or adverbs. The auxiliaries used are *chax* (will), *maixx* (not), and *daixx* (able).

Nouns may be used alone or serve as the head in noun phrases containing head, adjective(s), a numeral, and a classifier, in that order (Haas, 1964). The choice of classifiers is determined by the head noun. Thus, *khon* is the ordinary classifier for people; *lemxx* for sharp-pointed objects, such as knives, and for carts, books; *luukxx* for all kinds of balls, etc.

The English construction NP+be+adjective is taken up by the Thai NP+adjective or NP+intransitive verb. The first indicates that Thai discards the verb *be* when it is followed by an adjective. For example, a sentence *The school is big* is translated into Thai as *roongrian yaix* (literally, school big). The latter indicates that words corresponding to adjectives in English are true verbs in Thai. For example, a sentence *John is green* is translated into Thai as *jorn raihprasobxkarn* (literally, john inexperience).

Figure 3-6 shows a fragment of an ATN generating grammar. For simplicity, each grammar rule is written without arguments.

```
gen :- gen_sentence, print_sent.

gen_sentence :-
     get the semantic content of the case relation agent
     gen_np,                  % generate a NP describing the agent
     gen_vp.

gen_vp :-
     gen_vg,
     gen_vp_head.

gen_np :- gen_noun, gen_np2.
gen_np2 :- gen_adj, gen_np2.
gen_np2 :- gen_numeral, gen_np3.
gen_np2 :- gen_np3.
gen_np3 :- gen_npmods.
gen_numeral :- numeral, add classifier.

gen_vg :- add_passive_marker, gen_verb.

gen_vp_head :-
     get the semantic content of the case relation object
     gen_np,
     gen_vp_np.
gen_vp_np :-
     get the semantic content of the case relation recipient
     gen_np,
     gen_vp_mods.
gen_vp_np :- gen_vp_mods.

gen_vpmods :- gen_pp.
gen_vpmods :- null_gsr.

gen_pp :-  gen_prep, gen_np.

null_gsr.                % no further relation in the SR
```

Figure 3-6. A Fragment of an ATN Generating Grammar.

3.5.2 The Process of Generation

This section uses the example shown in Figure 3-4 and the ATN in Figure 3-6 to illustrate the generation process. The top level generator procedure, *gen*, is given as argument the semantic representation delivered from

the parser. *Gen* first initializes a variable *String* to hold the surface string being built. It then turns over control to the procedure *gen_sentence* which will looks at the generation of a normal subject-verb-object sentence. *Gen_sentence* first determines whether to generate a passive or an active sentence. Since an active flag is present, *gen_np* is called to express the contents of *agent*, *coach2_1*. This produces a surface string *kruufykx* (coach) which is added to *String* when control is returned to *gen_sentence*.

The process continues with a call to *gen_vp*. At this level, the input is *buy1_1* which contains the modality information such as aspect, voice, sentence negation, and modalization. Since the concept *buy1_1* is not negated, there is no modal concept, the active voice flag is present, and there is no aspect information, the verb group *syyx* (to buy) is generated as the main verb of the generated sentence.

Back in *gen_vp*, the verb group string is added to the *String*, which is now *kruufykx syyx*, and the process jumps to *gen_vp_head*. There, the call for an *object* case relation is found, and so the *gen_np* is called to generate a noun phrase for the contents of *object*, *ball2_1*.

At *gen_np*, the process starts at *gen_noun* where it generates the head noun string *luukxxborn* (ball), which is added to the end of *String* (previously empty). It then proceeds to *gen_adj* to generate a string of adjectives. At this point, the process checks to see if a case relation of type *attr* is present. If it is, the process returns a relation-value pair. Since *ball2_1* has associated with it an *attr*ibute concept *color*, whose value is *red1_1*, an adjective string *siihhdang* (red) is generated, and is added to *String*.

The generation process now determines that there is no further

relation of type *attr*, it jumps to the *gen_numeral* where the numeral string *samh luukxx* (numeral word + classifier) is generated. The numeral string *samh luukxx* is then added to *String*, which is now *luukxxborn siihhdang samh luukxx*. Next, the calls to *gen_np3* and then *gen_npmods* end the NP generation network. Control is back to *gen_vp_head* where the NP string is added to the top level *String*, which is now *kruufykx syyx luukxxborn siihhdang samh luukxx*.

The generation process then proceeds with *gen_vp_np* and jumps to *gen_vp_mods* emptying the input. It ends with the completed sentence: *kruufykx syyh luukxxborn siihhdang samh luukxx* (literally, coach buy ball red three ball).

3.6 A Complete Example

This section presents an example of translation from English to Thai on an input *The coach stabbed the man in the red coat with a knife.* Additional examples handled by the system are in Appendix C.

Once the preprocessor has taken place, the input to the parser is of the following form:

[the,coach,stabbed,the,man,in,the,red,coat,with,a,knife,.]

Figure 3-5 shows the parsing ATN grammar that is used to parse simple declarative sentences. This example, however, will not cover the details of syntactic parsing at the NP, VP_V and PP levels.

The first step in parsing is to parse a noun phrase. This parses the string *The coach*, and instantiates the variables *Det, Quant,AdjMods,* and

Noun to *def* (definiteness), *[]*, *[]*, and *coach* respectively. These variables are passed as arguments to the semantic procedure *build_np* where a semantic representation for a corresponding noun phrase is to be constructed. Since the lexical entry for *coach* carries two senses: *coach1* (a vehicle sense), and *coach2* (a person sense), instances for each sense, *coach1_1* and *coach2_1*, are created. These instance frames are represented as follows:

```
frame(coach1_1:[
                ref:[value:[def]],
                number:[value:[sin]],
                in_of:[value:[coach1]]]).

frame(coach2_1:[
                ref:[value:[def]],
                number:[value:[sin]],
                in_of:[value:[coach2]]]).
```

At this point *build_np* tries to disambiguate between *coach1_1* and *coach2_1* by checking the semantic compatibility between each noun sense instance and the constituents found in front of the head noun, in this case refers to a list of prenominal adjectival modifiers, *AdjMods*. Since *AdjMods* is empty, the appropriate sense cannot be selected. *Build_np* then creates *ambiguity_1*, an ambiguous concept that holds all possible sense instances, and transmits it as the *Subj* of the clause. The frame representation of *ambiguity_1* is shown below.

```
frame($ambiguity_1:[
                consists_of:[value:[coach1_1,coach2_1]],
                in_of:[value:[$ambiguity]]]).
```

The parser then discovers that *stabbed* is the main verb of the clause, and the lexicon carries only one sense for stab. The sense instance *stab1_1* is

then created and augmented with modality information such as tense, negation and voice. This structure is transmitted as the *Verb* of the clause.

```
frame(stab1_1:[
              negation:[value:[n]],
              tense:[value:[past]],
              voice:[value:[active]],
              in_of:[value:[stab1]]]).
```

The parser proceeds to parse *the man* as a NP, producing the noun sense instance *man1_1*, which is transmitted as *Obj* of the clause and whose representation is shown below:

```
frame(man1_1:[
              ref:[value:[def]],
              number:[value:[sin]],
              in_of:[value:[stab1]]]).
```

The parser next parses *in the red coat* as a PP, producing *pp_1*:

```
frame(pp_1:[
              prep:[value:[in]],
              prepobj:[value:[coat1_1]]]).

frame(coat1_1:[
              ref:[value:[def]],
              number:[value:[sin]],
              $color:[value:[red1_1]],
              in_of:[value:[coat1]]]).

frame(red1_1:[in_of:[value:[coat1]]]).
```

Since a PP in this position can be attached to either the main verb of the clause or the noun phrase preceding it, it is transmitted as simply a *PPMods* of the clause.

The parser next parses *with a knife* as a PP, producing *pp_2* which is

then added to *PPMods*:

```
frame(pp_2:[
            prep:[value:[with]],
            prepobj:[value:[knife1_1]]]).

frame(knife1_1:[
            ref:[value:[indef]],
            number:[value:[sin]],
            in_of:[value:[knife1]]]).
```

At this point the parser is at the end of the clause (and string). This signals the final semantic procedure *build_clause* to construct the semantic representation of the clause by checking that all necessary slot constraints are satisfied, and the conceptual case roles are filled accordingly.

Build_clause is handed certain variables: *Subj, Verb, Obj,* and *PPMods*, which are previously set to *ambiguity_1, stab1_1, man1_1,* and */pp_1,pp_2/* respectively. Since *stab1_1* allows a concept of type *person* as the filler of its *agent* role, *ambiguity_1* is disambiguated as *coach2_1*, and so the *agent* role of *stab1_1* is filled with *coach2_1*:

```
frame(stab1_1:[
            $agent:[value:[coach2_1]],
            negation:[value:[n]],
            tense:[value:[past]],
            voice:[value:[active]],
            in_of:[value:[stab1]]]).
```

Stab1_1 allows a concept of type *animate* as the filler of its *object* role, which *man1_1* belongs, and so its object role is filled with *man1_1*:

```
frame(stab1_1:[
            $object:[value:[man1_1]],
            $agent:[value:[coach2_1]],
            negation:[value:[n]],
```

```
                tense:[value:[past]],
                voice:[value:[active]],
                in_of:[value:[stab1]]]).
```

Next, the procedure *pp_attachment* is called to resolve the attachment ambiguity. In this stage, a basic loop is set around the contents of *PPMods*. The function of the loop is to attach a PP either to the noun preceding it, *man1_1*, or the main verb of the clause, *stab1_1*. To attach *pp_1* (in the red coat), the process first checks to see if *pp_1* is a possible descriptor of *man1_1* (the male person) by consulting the dictionary definition of *man1* for case preferences. Since its dictionary definition includes a *descriptive* case [in,[$clothing],$descriptive,in2] so *pp_1* is attached to *man1_1*.

```
frame(man1_1:[
                $descriptive:[value:[in2_1]],
                ref:[value:[def]],
                number:[value:[sin]],
                in_of:[value:[stab1]]]).

frame(in2_1:[
                $object:[value:[coat1_1]],
                in_of:[value:[in2]]]).
```

Next, the process attempts to attach *pp_2* (with a knife). The process begins by attempting to attach *pp_2* to the NP concept *man1_1*. Since *man1_1* has no case preferences, the process continues checking to see if *stab1_1* has any case preferences. *Stab1_1* has an instrument case [with,[$inst],$instrument,with1] that matches the preposition *with* and *knife1_1* to *with a knife*, so the PP is attached to *stab1_1*.

```
frame(stab1_1:[
                $instrument:[value:[knife1_1]],
                $object:[value:[man1_1]],
```

```
$agent:[value:[coach2_1]],
negation:[value:[n]],
tense:[value:[past]],
voice:[value:[active]],
in_of:[value:[stab1]]]).
```

The parsing process is now complete. It then transmits *stab1_1* as the semantic representation (SR) of the sentence *The coach stabbed the man in the red coat with a knife* to the generator.

The generator then takes as input the SR, uses the ATN generating grammar shown in Figure 3-6 to produce a Thai surface string. Since the generation process has been previously described in Section 3.5.2, the details at each level will be ignored.

The first step in the generation process is to look for the case relation (case role) *agent* in the concept *stab1_1*. The *agent* is found, and so is the concept *coach2_1*. *Gen_np* is then called to generate a noun phrase describing the concept *coach2_1*. At the end of *gen_np*, a surface string *kruufykx* (coach) is generated and added to the *String*, a variable which is used to collect the surface sentence being built.

At *gen_vp* the generator generates a verb string *thaeng* (to stab) to describe the concept *stab1_1*. It is then added to *String* which is now *kruufykx thaeng*. The generator then jumps to *gen_vp_head* where it generates an object noun phrase string *phuuxxchaai thiixx suamh syaxxklum siihhdang* (literally, man that wear coat red) describing the concept *man1_1*. Note that a relative clause transformation is applied when a PP is used to modify the noun preceding it. At end, this string is added to *String* which is now read *kruufykx thaeng phuuxxchaai thiixx suamh syaxxklum siihhdang*.

Next, the generator jumps to *gen_vpmods* where it calls *gen_pp* to generate a postverbal prepositional phrase, an instrument concept *knife1_1*. The call to *gen_prep* automatically inserts a preposition *duayxx* (with) using the information (encoded in the *makeprep* slot) supplied by the dictionary entry for *thaeng* (to stab). And the instrument concept *knife1_1* is expressed as a string *miidxx* (knife).

Since the SR has no further case relations, the generation process ends with the completed sentence:

kruufykx thaeng phuuxxchaai thiixx suamh syaxxklum siihhdang duayxx miidxx

Literally, coach stab man that wear coat red with knife.

This Thai translation is grammatical and carries the same meaning as the original English sentence. It is generated directly from the SR which indicates the past event consisting of the coach (a trainer) stabbing an object which is the man who wears the red coat, and using a knife as the instrument to stab. Note that the generation of Thai NP is different from English in that Thai adjectives always come after the head noun, the articles are entirely dropped and the postnominal prepositional phrase is a relative clause transformation in Thai. In addition, the fact that the SR is the past event could not be represented in Thai since an adverb of time which is used to indicate tense in Thai is not handled in this thesis.

CHAPTER 4

RESULTS

The system has been tested on a range of sentences with which it is equipped to deal. The quality of the translated sentences was evaluated only by the degree of *intelligibility*, that is, the degree to which the translated text can be understood by a native speaker of the target language (Nagao, Tsujii and Nakamura, 1988). Intelligibility is evaluated on a scale of 1 to 5; the categories are described below:

1. The meaning of the sentence is clear. Grammar and word usage are all appropriate, and no rewriting is needed.
2. The meaning of the sentence is clear, but there are some problems in grammar and word usage.
3. The basic thrust of the sentence is clear, but the evaluator is not sure of some parts because of grammar and word usage problems. The evaluator needs to clarify the meaning of those parts in the English original.
4. The sentence contains many grammatical and word usage problems, and retranslation is needed.
5. The sentence cannot be understood at all.

The evaluator uses the above scale to evaluate the output sentence without any reference made to the English original. This evaluation was done by a native Thai speaker on the Rochester Institute of Technology campus. Translating 56 English sentences to Thai gave the results shown in Figure 4-1.

Most of the sentences were translated intelligibly. Two sentences gave multiple translations, and there is no case where the translation was rated 5.

Intelligibility	Evaluation Results	
	Total (56)	Percent
1	50	89.28 %
2	4	5.36 %
3	1	3.57 %
4	1	1.79 %
5	0	0.00 %

Figure 4-1. Evaluation Results for Intelligibility.

Below are example sentences for each evaluation type. A complete listing of the input sentences, the output sentences, and their intelligibility results is given in Appendix C.

Intelligibility = 1

Sentence: John beats Mary with the bat.

Translation: jorn tii maeriixx duawxx maih

Literal Translation: john hit mary with bat

The Thai translation carries the same meaning as the original English sentence. The Thai verb *tii* is equivalent to the English verb *beat* which means *to strike repeatedly* and the noun *maih* expresses the instrument sense of the English noun *bat.* It is also grammatically correct. Its construction is equivalent to English. However, Thai drops articles entirely and the tense could not be represented since an adverb of time which is used to

indicate tense is not handled.

Intelligibility = 2

Sentence: The book was read by Mary.

Translation: nunghhsyyhh tuukx aaanx dooy maeriixx

Literal Translation: book be-en read by mary

The meaning of the translated sentence is clear but the passive construction is not acceptable. Thai normally uses the passive construction for the implication of something unpleasant, such as punishments, bad fortune, etc.

Intelligibility = 3

Sentence: The old man bought the boat.

Translation: phuxxchaai kaex syyh rya

Literal Translation: man old buy boat

It is understood that the man bought the boat but there is problem with the usage of the word *kaex* (old). The evaluator is not sure whether *kaex* is used to modify the noun *phuxxchaai* (man) or the main verb *syyh* (buy). When modifying the noun, its literal meaning is "the old man bought the boat", but when modifying the verb its literal meaning is "the man likes to buy the boat".

Intelligibility = 4

Sentence: John lost the ticket to Thailand.

Translation: jorn thamhaai tuahh pai praxtheedxxtai

Literal Translation: john lose ticket to Thailand

The generative grammar is currently unable to handle a double verb. In this case, the verb *thamhaai* (to lose) is a combination of two verbs *tham* (to make) and *haai* (to disappear); each carries its own full meaning. It has two functions; in one it is called *primary verb*, in the other *secondary verb*. When two verbs are used together as in the above sentence, the object, if any, usually comes between the primary and secondary verb. The correct translation would be *jorn tham tuahh pai praxtheedxxtai haai* (literally, john make ticket to Thailand disappear).

Intelligibility = 1; multiple translations

Sentence:	The coach is green.
Translation 1:	rodhmah siihhkhiaw
Literal Translation:	coach green
Translation 2:	kruufykx raihprasobxkarn
Literal Translation:	coach inexperience

The English verb *be* is disambiguated as "to have the property...". Since the system works on a single sentence and the results of applying selectional restrictions, two possible interpretations are produced. One reads the "person" sense of coach having the property "inexperience". Another reads the "vehicle" sense of coach having the color "green".

CHAPTER 5

CONCLUSIONS

This chapter sums up the work presented in this thesis, gives some directions for future research and concludes with a brief discussion of MT in general.

5.1 Summary of Work

This thesis has presented an implementation for interlingual translation between English and Thai. The system is a small-scale experimental system whose translation is not biased to any subject area, and translation is made without human intervention (automatic translation). It is capable of handling texts of simple syntactic constructs while performing in an ambiguous environment, both syntactically and semantically. The analysis lexicon currently used by the parser is not very big; however, it could be expanded easily. A complete processing cycle starts with sentence analysis (with emphasis on the problem of word-sense, prepositional phrase attachment, and referential ambiguity), goes through an unambiguous semantic representation equivalent to the meaning of the input text, and finally expresses that in Thai. Thus, automatic translation is effected. The system performs quite reliably, as the translated results show, over a range of sentences with which it is equipped to deal, and inappropriate choices of word-sense are fairly rare. However, it is felt that no major reorganization of the parsing process would

be necessary to recognize additional types of syntactic constructs.

During the parsing process, semantic judgement is incorporated in the process of syntactic recognition, thus allowing the syntactic and semantic processes to be carried out concurrently. The method used is to combine the syntactic recognition of a constituent with an evaluation of its semantic coherence as a unit and its compatibility with other units, based on global analysis of the contextual environment. In order to support this method, the notions of verb frame structures and the case preferences of prepositional phrase have been developed. These are static data structures embodying some of the semantic knowledge of the parser. The rest of this knowledge is distributed in the process of disambiguation, which is based on selectional restrictions. These control the application of semantic tests (judgements), which are guided both by the static knowledge of the parser and by the semantic content of the constituents. The tests evaluate contextual requirements within the constituent being analyzed, choose between alternative readings, and construct a semantic representation for the constituent.

The generation process explores the use of an ATN as a formalism for writing a generative grammar. The input to the grammar is the language independent semantic representation, delivered by the parser itself. The output is straightforward, simply concatenation onto the growing surface string. The grammar is designed specifically to generate Thai sentences. However, the same mechanism will be needed in order to generate sentences in languages other than Thai.

The most important deficiency of this system is the lack of knowledge to incorporate information from prior sentences. This naturally

introduces the problem of multi-sentence analysis, that is, computing the semantic consistency and coherence of a structure in relation to larger chunks of knowledge, and in the context of resolution of word-sense ambiguities, the issue of disambiguation by association. Until such knowledge is added, the system is bound to get many disambiguations wrong, because sentences can always be devised whose correct disambiguation depends on such knowledge.

The inability of the system to incorporate such knowledge and the fact that the system does nothing about inference-making processes and mechanisms, account for the less than robust approach to resolve the problem of the referential ambiguity. An attempt has been made only on pronoun resolution, and only three pronouns, *he*, *she* and *it* are known to the system. These are treated as though they are *the man*, *the woman*, and *the animal* or *the inanimate object*, respectively.

Parsing and generating require a lexicon. When an input string is presented to the parsing grammar, a search is first called to determine if all words in the string are known to the system. For unknown words, the system prints an error message and the program aborts. Since this system is just an experimental system, it is reasonable to admit failure. However, it might be interesting if the system could be equipped with the learning mechanisms so that it would constantly update itself.

As the evaluation results show, most test sentences were translated intelligibly. No misleading translations have been discovered, and it is felt that any Thai native speaker will understand this output in transliterated form, even though the word order or grammar might be "quaint".

5.2 Further Work

Following the summary already made in the previous section, it is obvious that the further work could be done to be in the area of linguistics (syntactic and semantic). At the lowest level, parsing program could be extended to accept complex sentences. Then, there is a certain issue to the resolution of the PP attachment ambiguity. The system in its current state will not parse a sentence like *John went to the store in the mall* if the generic information, encoded in the lexicon, requires attachment of *in the mall* to *the store.* One obvious solution to the problem is to modify the procedure *pp_attachment* to be able to check the immediately preceding object, which might be the prepositional object or any embedded clauses (besides the main verb of the sentence and the object of the main verb). This might be achieved in the following way (using the above sentence example): *pp_attachment* successfully returns the PP *in the mall* unattached, along with the prepositional object *the store* once it fails the attachment to *went.* Processing then continues by calling *pp_attachment* again. This time it would successfully attach *in the mall* to *the store.*

The interpretation of complex sentences may require a wider world knowledge. Extending the system to incorporate general world knowledge, along with the retrieval component are obviously a solution to correct interpretation.

Clearly, a lot of work remains to be done and many areas need to be investigated further. However, as the first step towards MT, this thesis has presented a translation system which involves semantic analysis, an area the MT community normally agrees that it is necessary to produce correct

translation.

5.3 Conclusion

MT faces serious problems in the area of semantics which prevent good quality mechanical translations. The use of linguistics tools such as semantic markers and selectional restrictions or case frames can only solve a certain class of semantic problems. It is, however, well known that a second certain class of meaning problems cannot be solved without considering knowledge of the world and, in many problems of lexical ambiguity, understanding of input texts. Currently there exist a number of experimental MT systems that aim to solve the problems. Though concrete solutions have not yet been defined, the results in translating texts in restricted domains seem promising. Because of such results, MT will not disappear from the scene and will not be declared impractical or impossible. On the contrary, it may prove worthwhile to design a computerized translation system to take advantage of these restrictions in a domain dependent system, if there is sufficient demand for translation in such a field.

BIBLIOGRAPHY

Bhandari, Archana. 1989. *Enhancements to the Frame Virtual Machine.* M.S. Thesis. Rochester Institute of Technology.

Campbell, Stuart, and Shaweevongs, Chuan. 1970. *The Fundamentals of the Thai Language.* Kent, England: Bailey Bros. & Swinfen Ltd.

Fillmore, Charles J. 1968. The case for Case. In *Universals in Linguistic Theory*, eds. Emmon Bach, and Robert T. Harms, 1-88, New York: Holt, Rinehart and Winston, Inc.

Haas, Mary R. 1964. *Thai-English Student's Dictionary.* Stanford, California: Stanford University Press.

Hiss, LaMora S. 1987. *A Frame Virtual Machine in C-Prolog.* M.S. Thesis. Rochester Institute of Technology.

Hutchins, W. J. 1986. *Machine Translation: Past, Present, Future.* West Sussex, England: Ellis Horwood Limited.

Minsky, Marvin. 1985. A Framework for Representing Knowledge. In *Readings in Knowledge Representations*, eds. Ronald J. Brachman, and Hector J. Levesque, 245-262. Los Altos, California: Morgan Kaufmann Publishers, Inc.

Nagao, Makoto, Tsujii, Jun-ichi, and Nakamura, Jun-ichi. 1988. The Japanese Government Project for Machine Translation. In *Machine Translation Systems*, ed. Jonathan Slocum, 141-186. Cambridge: England: the Press Syndicate of the University of Cambridge.

Nirenburg, Sergei (Ed.) 1987. *Machine Translation: Theoretical and Methodological Issues.* Cambridge: Cambridge University Press.

Schank, Roger C. 1973. Identification of Conceptualizations Underlying Natural Language. In *Computer Models of Thought and Language,* eds. Roger C. Schank, and Kenneth Mark Colby, 187-247. San Francisco: W. H. Freeman and Company.

Shapiro, Stuart C. 1982. Generalization Augmented Transition Network Grammars for Generation from Semantic Networks. *American Journal of Computational Linguistics* 8:12-25.

Warotamasikkhadit, Udom. 1972. *Thai Syntax, an Outline.* Hungary: Mouton & Co. N.V.

Wilks, Yorick. 1975. An Intelligent Analyzer and Understander of English. *Communications of the ACM* 18:264-274.

Wilks, Yorick. 1979. Machine Translation and Artificial Intelligence. In *Translating and the Computer*, ed. Barbara M. Snell, 27-43. Amsterdam: North-Holland Publishing Company.

Wilks, Yorick, Huang, Xiuming, and Fass, Dan. 1985. *Syntax, Preference and Right Attachment.* Technical Report, Computing Research Laboratory, New Mexico State University, New Mexico.

Woods, William A. 1970. Transition Network Grammars for Natural Language Analysis. *Computational Linguistics* 10:591-606.

APPENDIX A

TRANSLITERATION SYSTEM

The transliteration system used in this thesis is based on Hass (Haas, 1964). However, an adaptation was made, due to the limitation of the characters on the keyboards and printers.

A.1 Consonants

Thai has 44 consonants, which are transliterated as follows:

ก	k	ข	kh	ฃ	kh	ค	kh	ฅ	kh
ฆ	kh	ง	ng	จ	c	ฉ	ch	ช	ch
ซ	s	ฌ	ch	ญ	y	ฎ	d	ฏ	t
ฐ	th	ฑ	th	ฒ	th	ณ	n	ด	d
ต	t	ถ	th	ท	th	ธ	th	น	n
บ	b	ป	p	ผ	ph	ฝ	f	พ	ph
ฟ	f	ภ	ph	ม	m	ย	y	ร	r
ล	l	ว	w	ศ	s	ษ	s	ส	s
ห	h	ฬ	l	อ	a	ฮ	h		

A.2 Vowels

Vowels are written either in front of, above, below, or all around the consonants to which they refer. Below shows 32 basic vowels which are used with consonants to form words. The first column is the actual Thai writing, and the second column is the transliteration. Since a vowel must always be used with a consonant, the consonant อ is added to support the vowel and in this case the อ is always silent.

Short Vowel		Long Vowel	
อะ	a	อา	i
อิ	æa	อี	ii
อึ	y	อื	yy
อุ	u	อู	uu
เอะ	e	เอ	ee
แอะ	æ	แอ	æ
โอะ	o	โอ	oo
เอาะ	or	ออ	or
เออะ	oe	เออ	oe
เอียะ	ia	เอีย	ia
เอือะ	ya	เอือ	ya
อัวะ	ua	อัว	ua
ฤ	ri, ry, roe	ฤา	ryy
ฦ	li, ly, loe	ฦา	lyy
อำ	am		
ไอ	ai		
ใอ	ai		
เอา	aw		

A.3 Tones

The Thai language is a *tonal language* (Campbell and Shaweevongs, 1970), which means that a word may have two or more distinct and quite unrelated meanings, depending on the tone with which it is pronounced. Each syllable (and word) has one of the five tones: middle, low, falling, high, and rising. To denote tones, letter(s) are attached to each syllable: none for the middle tone, 'x' for the low, 'xx' for the falling, 'h' for the high, and 'hh' for the rising tone. Below shows an example used for each tone:

Tone	Example
Mid	khaa (business)
Low	khaax (to be embedded)
Fall	khaaxx (a spice)
High	khaah (leg)
Rising	khaahh (to kill)

A.4 Ambiguity in Actual Thai Writing

As presented in this thesis, the words in each translated sentence are written with spaces between them. Ordinarily however, passages written in Thai are run together results in ambiguous spelling as well as meaning. Typical example is a word *taaglom* which could be translated in English as a verb meaning "expose to air" *taag_lom* (literally, expose air) or as a noun meaning "round eyes" *taa_glom* (literally, eye round).

Thai makes no use of inflections. Therefore, there is only one form for verbs and for nouns. For example, a Thai verb like *kin* can mean "eat,

eats, eating, ate, eaten, to eat". In the same way, a Thai noun *rya* can correspond to two forms of English noun "boat, boats". However, this does not mean that Thai is not as definite and precise as English; it is just that when such precision is unnecessary, Thai is not bound, as English, to be precise about it. Thus, the meaning content of such a phrase as "Horses like to run" is not different from that of the phrase "A horse likes to run". Also, Thai drops the articles (a, an, the) entirely. Thus, in translating from Thai to English, one should use both the context and the recognition of special words in order to arrive at the correct meaning.

APPENDIX B

THE PARSER'S VOCABULARY

ADJECTIVES

green(2)	old(*)	red(1)	ripe(2)	

COMMON NOUNS

ball(2)	banana(1)	bank(2)	bat(2)	bed(2)
boat(1)	book(**)	can(**)	car(1)	club(2)
coach(2)	coat(**)	dinner(1)	dog(1)	dress(1)
fare(1)	game(1)	hand(**)	house(1)	knife(1)
man(**)	monkey(1)	old(*)	park(**)	pill(1)
rack(1)	school(1)	stamp(2)	star(2)	ticket(1)
train(**)	truth(1)			

PROPER NAMES

John(1)	Mary(1)	Thai(2)	Thailand(1)

PREPOSITIONS

by	for	in	on	to
with				

VERBS

admit(2)	ask(2)	be(2)	beat(2)	buy(1)
call(2)	collect(2)	give(1)	go(1)	have(1)
kick(1)	lose(2)	man(**)	marry(1)	position(1)
read(1)	run(1)	see(1)	serve(2)	stab(1)
swim(1)	take(4)	walk(1)	wear(1)	

DETERMINERS

a	the

QUANTIFIER

three

PRONOUNS

he	it	she

MODAL VERBS

can(**)	will

n the number of word senses
* used both as an adjective and a noun
** used both as a verb and a noun

APPENDIX C

ETRANS TRANSLATION

This appendix first shows a sample run of translation, which starts with the source text, shows the internal semantic representation, the corresponding target text, and concludes with runtime statistics. For the sake of readability, the output from the Frame Package has been deleted, and the translated test sentences are listed in brief. Each sentence shows the source text, the target text, runtime statistics, and its evaluation result.

C.1 Sample Run

```
pith-nxw4544[4] prolog etrans
C-Prolog version 1.4
[ Restoring file etrans ]
[ Expanding atom space from 128K to 150K ]
[ Expanding heap from 256K to 516K ]

yes
| ?- trans.

Enter text:
|: The coach stabbed the man in the red coat with a knife.

knife1_1:
    [
    ref:
        value:[indef]              indefinite article
    number:
        value:[sin]                singular
    in_of:
        value:[knife1]             knife1: an instrument for cutting
    ]
```

```
red1_2:
    [
    in_of:
        value:[red1]                    red1: color red
    ]

coat1_1:
    [
    ref:
        value:[def]
    number:
        value:[sin]
    $color:                             $color is an attribute name
        value:[red1_2]                  red1_2 is an attribute value
    in_of:
        value:[coat1]                   coat1: an outer garment
    ]

in2_1:
    [
    $object:
        value:[coat1_1]
    in_of:
        value:[in2]                     in2: to wear
    ]

man1_1:
    [
    $descriptive:                       descriptive is a case which is used for
        value:[in2_1]                   prepositional phrases which modify nouns
    ref:
        value:[def]
    number:
        value:[sin]
    in_of:
        value:[man1]                    man1: a noun meaning a male person
    ]

coach2_2:
    [
    ref:
        value:[def]
    number:
        value:[sin]
    in_of:
        value:[coach2]                  coach2: a trainer
    ]
```

```
stab1_1:
    [
    $instrument:                    the instrument used to stab
        value:[knife1_1]
    $object:                        the one who is stabbed
        value:[man1_1]
    $agent:                         the one who does the stabbing
        value:[coach2_2]
    negation:
        value:[n]                   the sentence is not negated
    tense:
        value:[past]                past event
    voice:
        value:[active]              active construction
    in_of:
        value:[stab1]               stab1: to wound someone with a pointed weapon
    ]
```

Translation:
kruufykx thaeng phuxxchaai thiixx suamh syaxxkhlum siihhdang duayxx miidxx

```
atom space: 150K (116552 bytes used)
aux. stack: 8K (0 bytes used)
trail: 64K (2036 bytes used)
heap: 516K (485052 bytes used)
global stack: 256K (138208 bytes used)
local stack: 128K (3076 bytes used)
Runtime:  200.98 sec.


Enter text:
|: abort.

-====== PROGRAM TERMINATED ======-

yes
| ?- halt.

[ Prolog execution halted ]
```

The above Thai translation has an intelligibility score of 1. The sentence is grammatical and carries the same meaning as the original English sentence. The sentence, whose literal translation is "coach stab man that wear coat red with knife", is constructed directly from the semantic representation which indicates the past event consisting of the coach (a trainer) stabbing an object which is the man who wears the red coat using the instrument which is the knife.

C.2 Test Sentences

Sentence: John took a pill.
Translation: jorn kin ya
Literal Translation: john eat pill
Runtime: 109.23 sec
Intelligibility: 1

Sentence: John took a pill for Mary.
Translation: jorn khamooy ya haixx maeriixx
Literal Translation: john steal pill for Mary
Runtime: 118.93 sec
Intelligibility: 1

Sentence: John took Mary.
Translation: jorn koong maeriixx
Literal Translation: john swindle Mary
Runtime: 63.85 sec
Intelligibility: 1

Sentence: John took Mary to the ball.
Translation: jorn phaa maeriixx pai ngaanborn
Literal Translation: john bring mary to ball
Runtime: 96.50 sec
Intelligibility: 1
The verb *phaa* is selected to translate a ptrans sense of *take* in this case because we are referring to taking a person somewhere.

Sentence: John took the ball.
Translation: jorn khamooy luukxxborn
Literal Translation: john steal ball
Runtime: 115.88 sec
Intelligibility: 1

Sentence: John took the ball to school.
Translation: jorn aaw luukxxborn pai roongrian
Literal Translation: john bring ball to school
Runtime: 180.27 sec
Intelligibility: 1
The verb *aaw* is selected to translate a ptrans sense of *take* in this case because we are referring to taking an inanimate object somewhere.

Sentence: John took a train to school.
Translation: jorn dooysarn rodhphai pai roongrian
Literal Translation: john take train to school
Runtime: 85.75 sec
Intelligibility: 1
The verb *dooysarn* is selected to translate a ptrans sense of *take* in this case, because we are referring to taking some kinds of transportation somewhere.

Sentence: John positioned the dress on the rack.
Translation: jorn chiih syaxxkraploongchud bon raaw
Literal Translation: john position dress on rack
Runtime: 61.28 sec
Intelligibility: 1

Sentence: John wanted the dress on the rack.
Translation: jorn torngkarn syaxxkraploongchud samhrubx maeriixx
Literal Translation: john want dress on rack
Runtime: 70.12 sec
Intelligibility: 1

Sentence: John wanted the dress for Mary.
Translation: jorn torngkarn syaxxkraploongchud thiixx yuu bon raaw
Literal Translation: john want dress that locate on rack
Runtime: 49.03 sec
Intelligibility: 1

Sentence: John bought the house for Mary.
Translation: jorn syyh baanxx haixx maeriixx
Literal Translation: john buy house for mary
Runtime: 51.00 sec
Intelligibility: 1

Sentence: John bought the house by the school.
Translation: jorn syyh baanxx thiixx yuu khaangxx roongrian
Literal Translation: john buy house that locate by school
Runtime: 72.03 sec
Intelligibility: 1

Sentence: John bought the ball by the school.
Translation: jorn syyh luukxxborn khaangxx roongrian
Literal Translation: john buy ball by school
Runtime: 84.73 sec
Intelligibility: 2
In this context, the verb *syyh* (to buy) requires that the preposition *jaak* (from) must be used to indicate that the thing is being bought from somewhere.

Sentence: The old man bought the boat.
Translation: phuxxchaai kaex syyh rya
Literal Translation: man old buy boat
Runtime: 55.10 sec
Intelligibility: 3
It is understood that the man bought the boat but there is problem with the usage of the word *kaex* (old). The evaluator is not sure whether *kaex* is used to modify the noun *phuxxchaai* (man) or the main verb *syyh* (buy). When modifying the noun, its literal meaning is "the old man bought the boat", but when modifying the verb its literal meaning is "the man likes to buy the

boat".

Sentence: The old man the boat.
Translation: khonkaex khub rya
Literal Translation: old drive boat
Runtime: 59.32 sec
Intelligibility: 1

Sentence: John lost the game to Mary.
Translation: jorn phaeh karnkhaengkhung kae maeriixx
Literal Translation: john lose game to mary
Runtime: 75.62 sec
Intelligibility: 1

Sentence: John lost the ticket to Thailand.
Translation: jorn thamhaai tuahh pai praxtheedxxtai
Literal Translation: john lose ticket to Thailand
Runtime: 86.90 sec
Intelligibility: 4
This is a case of the double verb usage in Thai. The verb *thamhaai* (to lose) is a combination of two verbs *tham* (to make) and *haai* (to disappear); each carries its own full meaning. It has two functions in one of which it is called *primary verb*, in the other *secondary verb*. When two verbs are used together as in the above sentence, the object, if any, usually comes between the primary and secondary verb. The correct translation would be *jorn tham tuahh pai praxtheedxxtai haai* (literally, john make ticket to Thailand disappear).

Sentence: John has three red balls.
Translation: jorn mii luukxxborn samhh luukxx
Literal Translation: john have ball three ball
Runtime: 65.62 sec
Intelligibility: 1

Sentence: The coach married the star.
Translation: kruufykx tengxngaan daaraa
Literal Translation: coach marry star
Runtime: 132.18 sec
Intelligibility: 2
A preposition *kub* (with) must follow the verb *tengxngaan* (to marry) to introduce an object.

Sentence: The coach kicked the ball.
Translation: kruufykx tex luukxxborn
Literal Translation: coach kick ball
Runtime: 131.68 sec
Intelligibility: 1

Sentence: The man in the red coat kicked the ball.
Translation: phuxxchaai thiixx suamh syaxxkhlum siihhdang tex luukxxborn
Literal Translation: man that wear coat red kick ball
Runtime: 117.35 sec
Intelligibility: 1

Sentence: John read the book by Mary.
Translation: jorn aaanx nunghhsyyhh thiixx khianhhdooy maeriixx
Literal Translation: john read book that write by mary
Runtime: 53.22 sec
Intelligibility: 1

Sentence: The book was read by Mary.
Translation: nunghhsyyhh tuukx aaanx dooy maeriixx
Literal Translation: book be read by mary
Runtime: 43.65 sec
Intelligibility: 2
The Thai translation carries the same meaning as the original English sentence. However, the active construction is preferred since Thai uses the passive construction only when referring to events of a violent or unpleasant nature.

Sentence: John read the book on Thailand.
Translation: jorn aaanx nunghhsyyhh thiixx kiawxkub praxtheedxxtai
Literal Translation: john read book that about Thailand
Runtime: 51.95 sec
Intelligibility: 1

Sentence: John can read Thai.
Translation: jorn aaanx phaasaahhtai daixx
Literal Translation: john read Thai able
Runtime: 41.80 sec
Intelligibility: 1

Sentence: John can't read Thai.
Translation: jorn aaanx phaasaahhtai maixx daixx
Literal Translation: john read Thai not able
Runtime: 41.75 sec
Intelligibility: 1

Sentence: John ran the school.
Translation: jorn borihaanhh roongrian
Literal Translation: john manage school
Runtime: 57.42 sec
Intelligibility: 1

Sentence:	John ran by the school.
Translation:	jorn wingxx phaanx roongrian
Literal Translation:	john run by school
Runtime:	68.83 sec
Intelligibility:	1

Sentence:	John walked in the park.
Translation:	jorn doen nai suansaatharana
Literal Translation:	John walk in park
Runtime:	59.48 sec
Intelligibility:	1

Sentence:	John walked the dog.
Translation:	jorn doencuung mahh
Literal Translation:	John walk dog
Runtime:	49.53 sec
Intelligibility:	1

Sentence:	John went to the ball with Mary.
Translation:	jorn pai ngaanborn kub maeriixx
Literal Translation:	John go ball with Mary
Runtime:	66.87 sec
Intelligibility:	1

Sentence:	John went to school with Mary.
Translation:	jorn pai roongrian kub maeriixx
Literal Translation:	John go school with mary
Runtime:	53.92 sec
Intelligibility:	1

Sentence:	John went to school by train.
Translation:	jorn pai roongrian dooythaang rodhphai
Literal Translation:	john go school by train
Runtime:	59.32 sec
Intelligibility:	1

Sentence:	John went by train.
Translation:	jorn pai dooythaang rodhphai
Literal Translation:	John go by train
Runtime:	41.25 sec
Intelligibility:	1

Sentence:	John swam to the bank.
Translation:	jorn waaynumh pai chaayphungx
Literal Translation:	john swim to bank
Runtime:	63.82 sec
Intelligibility:	1

Sentence: John gave a bed to Mary.
Translation: jorn haixx tiangnorn kae maeriixx
Literal Translation: John give bed to Mary
Runtime: 75.22 sec
Intelligibility: 1

Sentence: John handed Mary a red ball.
Translation: jorn songxx luukxxborn siihhdang haixx maeriixx
Literal Translation: John hand ball red to Mary
Runtime: 96.90 sec
Intelligibility: 1

Sentence: John was admitted to the club.
Translation: jorn tuukx rubhkhawwxx samoosornghh
Literal Translation: John be admit club
Runtime: 71.67 sec
Intelligibility: 2
The Thai translation carries the same meaning as the original English sentence. However, the active construction is preferred since Thai uses the passive construction only when referring to events of a violent or unpleasant nature.

Sentence: John admitted the truth.
Translation: jorn yormrub klwaamcing
Literal Translation: John admit truth
Runtime: 59.22 sec
Intelligibility: 1

Sentence: John asked Mary a question.
Translation: jorn thaamhh punhahh jaak maeriixx
Literal Translation: John ask question from Mary
Runtime: 66.77 sec
Intelligibility: 1

Sentence: John asked a question.
Translation: jorn thaamhh punhahh
Literal Translation: John ask question
Runtime: 63.25 sec
Intelligibility: 1

Sentence: John asked for the club.
Translation: jorn khorh maihgolb
Literal Translation: John ask club
Runtime: 63.90 sec
Intelligibility: 1

Sentence: John called Mary.
Translation: jorn riak maeriixx
Literal Translation: John call Mary
Runtime: 42.70 sec
Intelligibility: 1

Sentence: John called on Mary.
Translation: jorn paiyiamxx maeriixx
Literal Translation: John visit Mary
Runtime: 45.43 sec
Intelligibility: 1

Sentence: John collected fares.
Translation: jorn kebx khaaxxdooysaarn
Literal Translation: John collect fare
Runtime: 60.80 sec
Intelligibility: 1

Sentence: John will collect stamps.
Translation: jorn cha sasom duangtraapraisanee
Literal Translation: John will collect stamp
Runtime: 76.97 sec
Intelligibility: 1

Sentence: John served dinner.
Translation: jorn borikarn aaharnkhamxx
Literal translation: John serve dinner
Runtime: 90.93 sec
Intelligibility: 1

Sentence: John served the ball.
Translation: jorn serbx luukxxborn
Literal Translation: John serve ball
Runtime: 95.92 sec
Intelligibility: 1

Sentence: The coach is green.
Translation 1: rodhmah siihhkhiaw
Literal Translation: coach green
Translation 2: kruufykx raihprasobxkarn
Literal Translation: coach inexperience
Runtime: 99.65 sec
Intelligibility: 1

Sentence: The car is green.
Translation: rodh siihhkhiaw
Literal Translation: car green
Runtime: 54.10 sec
Intelligibility: 1

Sentence:	John beats Mary.
Translation 1:	jorn chana maeriixx
Literal Translation:	John defeat Mary
Translation 2:	jorn tii maeriixx
Literal Translation:	John hit Mary
Runtime:	64.17 sec
Intelligibility:	1

Sentence:	John beats Mary with the bat.
Translation:	jorn tii maeriixx duawxx maih
Literal Translation:	John hit Mary with bat
Runtime:	84.42 sec
Intelligibility:	1

Sentence:	John stabbed the man in the red coat with a knife.
Translation:	jorn thaeng phuxxchaai thiixx suamh syaxxklum siihhdang duawxx miidxx
Literal Translation:	John stab man that wear coat red with knife
Runtime:	200.98 sec
Intelligibility:	1

Sentence:	John saw the man in the red coat.
Translation:	jorn henh phuxxchaai thiixx suamh syaxxkhlum siihhdang
Literal Translation:	John see man that wear coat red
Runtime:	94.12 sec
Intelligibility:	1

Sentences:	John gave a banana to the monkey. It was ripe.
Translations:	jorn haixx kluayxx kae ling kluayxx suk
Literal Translations:	John give banana to monkey banana ripe
Runtimes:	107.13 sec 156.52 sec
Intelligibility:	1

This is a case of resolving pronoun reference. Since the parser successfully disambiguate the pronoun *it* as *banana* rather than *monkey*, the translated output is generated to show what resolution decision has been made. No attempt is made to use a pronoun for a re-expressed object.

Sentences:	John walked to school with Mary.
	He wore the green coat.
	She wore the red dress.
Translations:	jorn doen pai roongrian kub maeriixx
	jorn suamh syaxxklum siihhkhiaw
	maeriixx suamh syaxxkraploongchud siihhdang
Literal Translations:	John walk to school with Mary
	John wear coat green
	Mary wear dress red
Runtimes:	69.83 sec
	116.60 sec
	164.73 sec
Intelligibility:	1