An Active Bilingual Lexicon for Machine Translation

Igal GOLAN, Shalom LAPPIN*, Mori RIMON

IBM Israel Scientific Center,
The Technion City, Haifa 32000, Israel

(*) Main affiliation: The University of Haifa

E-mail address: golan and/or lappin and/or rimon @israearn.bitnet

Abstract

An approach to the Transfer phase of a Machine Translation system is presented, where the bilingual lexicon plays an active role, guiding Transfer by means of executable descriptions of word senses. The means for lexical sense specification are, however, general enough and can in principle apply to other system architectures, e.g. in the Generation phase if Transfer is intentionally kept minimal. The active lexicon is the one and only system component which is exposed to users and can serve to linguistically control Transfer effects. A unified approach to lexicon creation and maintenance is proposed, which contains means to gradually refine sense specification and tailor the definitions to specific text domains. The underlying linguistic principles, the nature of sense distinction required for translation, and the formal structure of the lexicon are discussed.

1. Introduction

While methods of monolingual Analysis and Generation are also treated in other contexts, bilingual Transfer problems are hardly investigated outside the context of Machine Translation. Research in Machine Translation can, in this case, make a specific contribution to Computational Linguistics. The general issue here is the formal representation of phrase structures and lexical units and the methodology for specifying transformations between these representations in two (or more) languages. The role of the bilingual lexicon in the Transfer activity, and its power to assist in the resolution of mapping problems, is a key element.

In this paper we present an approach to the formal representation of bilingual lexical knowledge and to the way this knowledge is incorporated into the translation process. In section 2 we describe the role and place of the bilingual lexicon in the translation process, present the concept of executable descriptions of word senses as lexical definitions, and discuss some aspects of practical usage. Our approach to the sense distinction required for translation, which is different from monolingual sense distinction, is discussed in section 3. In section 4 we make a few methodological comments, arguing that what is often portrayed as the ideal Transfer-based architecture, is not the only, and not necessarily the best way to achieve modularity and save work. Section 5 contains a formal definition of the lexicon specification language with some discussion of its features and the intended restrictions on the power given to the lexicographer. Finally, an additional example is given in detail in section 6.

This work has been carried out as part of the MENTOR project, where several groups in European IBM Scientific Centers are collaborating on M(A)T research. The approach presented here has been developed and prototyped by the group in Haifa, Israel, as part of the proposal for the design of Transfer-related operations. The examples below involve translations from English into Hebrew.

We thank our colleagues Danit Ben-Ari, Esther Bentur and Maria Vilkuna for their contributions and comments.

2. The Role and Content of the Bilingual Lexicon

[Cullingford 87] describes an MT system which is purely lexicon driven. His system follows the Conceptual Processing model, and is not Transfer-based, hence the emphasis there is on deep Analysis and Generation. Many other systems distinguish between Lexical Transfer and Structural Transfer, but they take different approaches to the actual separation of these two sub-processes. In the work reported here, an attempt was made to strictly separate lexicon-driven selection of target language equivalents from the global mapping of syntactic structures in the SL[1] into those of the TL (cf. [Biewer 85]). The lexicon lookup phase, which takes place before phrase structure transformations, gets as its input the internal data representation provided by the SL parser (PEG [Jensen 86], in our prototype). The terminal nodes (leaves of a parse tree) are searched in a pre-defined order for certain parts of speech. For each word in turn a target equivalent is selected from the bilingual lexicon and attached to the corresponding node in the parse. Features may also be added to other affected nodes. However, no structural modifications are made. Structural transformations are carried out as an independent sub-process, upon completion of the bilingual lexical phase, and are not discussed in this paper.

Since in many cases, and in fact for most verbs, several alternative translations exist, the selection is done by lexical differentiation rules. These rules refer to the syntactic environment of the word in the parse tree and to a limited number of semantic features. The rules can access any node and attribute identified by the parser. Given that the rules are stated in terms of the SL phrase structure, it seems more natural to apply them as close to Analysis as possible. Nevertheless, the sense distinction cannot be done as part of SL Analysis itself, as in many cases it depends on factors which may vary from one TL to another.

The sub-process of bilingual lexical substitution proceeds unidirectionally. No iterations take place for any given phrase. In some cases this may require extensive searching of the phrase

---

1 The following abbreviations are used throughout this paper:
SL = Source Language, BL = BiLingual, TL = Target Language

structure in a manner determined by the relevant differentiation rules.

The differentiation rules which comprise an entry in the bilingual lexicon are stated in a special language, and are directly executable by a set of functions (LISP, in our implementation). Although the phrase attributes and features they can refer to are similar, to some extent, to what is done in certain other systems (see for example [Nagao 86]), it seems that action rules are more flexible and powerful than static form-oriented structures. In fact, each statement in the set of rules which comprises a given lexical entry defines a correspondence between a syntactic environment (with semantic decoration) in the source language and a translation into the target language.

Consider, for example, one of the four rules which comprise the entry for the verb "avoid" in the English-Hebrew lexicon. This particular rule identifies the case where the current verb node (CN) has a postmodifier noun phrase and the head noun of that noun phrase has the semantic attribute Animate (e.g. "She avoids her old father."). In that case the Hebrew translation should be "התחמק מ". In terms of the lexicon specification language and the attributes used by the PEG parser, the rule looks as follows:

```
If ( CN has Postbrother ((CAT NP)(ANIMATE +)) )
   Then < Put (HEB "התחמק מ" );> ;
```

The *put* instruction attaches the Hebrew translation as a feature on the current node (CN is the default node assumed when no explicit node name is specified for *put*).

Certain lexical rules can be simply and elegantly formulated in terms of functions which identify grammatical relations of various kinds. Thus, for example, the rule for translating the verb "return" can be specified in terms of the feature Passive, and the presence or absence of a direct object. The set of rules which follows makes the required distinction between the ergative (intransitive) sense and the transitive reading of the verb:

```
Start RETURN VERB
1  If (call OBJECT) Then < Put (HEB "החזיר" );end;> ;
   @  Example: Ron returned the book.
   @  Example: The book was returned.
2  Put (HEB "חזר" );
   @  (this is the default translation)
   @  Example: Ron returned early.
Finish
```

Ideally, the feature OBJECT could be defined as part of the information provided by the parser. However, if for some reason the parser does not identify such relations, it can be defined as a function, written in the lexicon specification language described in this paper, which will evaluate the parse tree during Transfer. An example of such a function, which is included as a utility function in the active bilingual lexicon system, is given in [Golan 88], an extended version of this paper. This function tests a tree for the presence of a constituent which is the direct object of a given verb. If the verb appears in the passive voice, it assumes existence of a direct object (note that the active voice is selected for the Hebrew equivalent even in the passive case; passivization is done later in Generation).

It may be necessary to define functions which identify more complex grammatical relations than the direct object of a given verb. It is, for example, necessary to refer to the relation 'head noun argument, with property P, of an adjective', when formulating the rule for an adjective like "available":

```
Start AVAILABLE ADJ
If (Call N-of-ADJ (HUMAN))
  Then < Put (HEB "פנוי" );>
  @  Example: The doctor is not available now.
  @  Example: Any available salesperson can help you.
  Else < Put (HEB "מצוי" );> ;
  @  Example: This book is available in every store.
Finish
```

The algorithm N-of-ADJ, given in detail in section 6 below, characterizes a parameterized function which identifies the head noun argument, with property P, of a given adjective. This algorithm is written in the lexicon specification language and is also included as a utility function in the lexicon system. Rules which employ this function can render the translation of an adjective dependent upon the modified noun phrase having specific attributes. It should also be noted that the function N-of-ADJ identifies the noun when the adjective acts as a prenominal modifier as well as when it appears in the VP predicative form.

It is important to recognize that functions of this kind are useful to the extent that they can be employed in a significant number of lexical rules. There are, however, many cases in which the translation rules must be formulated directly as search procedures on trees. There are no obvious general grammatical functions which can be used to define the syntactic relations that must be tested in order to determine translation in these cases. Consider, for example, the set of rules for the adverb "precisely":

```
Start PRECISELY ADVERB
1  If (CN has ancestor((CAT NP PP)))
     Then < Put (HEB "בדיוק" );end;> ;
   @  Example: This is precisely the question.
   @  Example: Rina spoke precisely to the point.
2  Put (HEB "במדויק" );
   @  Example: Ron formulated his answer precisely.
Finish
```

The disjunctive relation of being the descendent of either an NP or a PP does not seem to be the sort of grammatical relation which one would wish to encode in a general grammatical function.

Similarly, consider the set of rules below, which are part of the definition of the verb "allow". The case where this verb appears in an active voice and forms part of the verb-particle construction "allow for" are identified by searching the tree for the preposition "for" which may be the head of a possibly non-adjacent PP:

```
1  If ( CN has First Postbrother (name-1((CAT PP))
     has Son name-2((CAT PREP)(base "for"))))
     Then < Put (HEB "לקח בחשבון" );
            Put in name-2 (HEB "");end;> ;
   @  Example: Ron allowed for the results.

2  If ( CN has Postbrother (A((CAT AVP))
     has Descendent (name-1((CAT PREP)(base "FOR"))
       before B((CAT NP)))))
     Then < Put (HEB "לקח בחשבון" );
            Put in name-1 (HEB "");end;> ;
   @  Example: Ron allowed reluctantly and cautiously
   @  for the results.
```

In fact, the basic need to look for a non-adjacent particle is rather common. It may justify inclusion of generalized

relations in the lexicon formalism. A generalized notion of "postbrother" will enable the statement of the sense distinction above as one rule. However, the detailed design of such generalized relations is very complex and is not yet included in our model.

The "allow for" example presents yet another problem for translation. The Hebrew translation "לקח בחשבון" consists of two words which have individual verb and noun entries in the TL lexicon. If the target translation is a recognized collocation or idiom it can be specified as such in the TL lexicon. But there are cases where an SL entry must be rephrased in the TL as an ad-hoc combination of words which does not form a lexical entity. In that case the lexical rule must mark words with corresponding parts of speech, or provide alternative information which will serve later in Generation to properly decide on the required morphological manipulation.

Generally, our experiments with actual lexical entries lead us to believe that the most efficient strategy for formulating lexical rules is to use a combination of general functions and rule-specific search procedures. A formal description of the specification language and a detailed example of its expressive power are given in sections 5 and 6 below.

The creation of a large scale full-fledged BL lexicon is a very heavy task, impeding development of product level systems. There is no way to avoid it, but there are ways to facilitate the practical development of the lexicon by making the process more modular. The scheme presented in this paper enables the system lexicographers - and each individual user - to proceed in steps. As a first, rough, approximation, one may simply define one "default translation"; e.g., for the verb "type":

Start TYPE VERB Put (HEB "הדפיס"); Finish

This will account for the use of "type" in the sense of "typewrite". Later, other senses can be defined (e.g. "טיווג", for "classify"), going from the most frequent in a given context to the relatively rare ones, thus achieving an increasingly refined level of sense specification. The optimal level of detail may depend on user needs and on specific text domains. In fact, different sense distinctions may be required for different domains. The user can modify the lexicon accordingly, or define his/her own private addenda lexicon, overriding identical entries in the main lexicon.

The lexicon of the system is accessible to revision by users. However, users cannot modify the global transformations which map syntactic structures in the SL into those of the TL, nor can they interfere with morphological aspects of Generation (TL values in the bilingual lexicon are given in a canonical form; e.g. for Hebrew verbs: 3rd person - singular - past tense - active voice). Consequently, user influence is constrained in such a way that it cannot disrupt the functioning of the large scale architecture of the system, but can at most affect local changes of a lexically based nature.

This is one obvious advantage of the strict separation between bilingual lexical transfer and general phrase structure transformations. Another benefit of this approach is the restriction of the structures that must be considered for transformations to standard, non-distorted SL phrase structures, as provided by the SL parser. This eliminates complications and loss of generality of the structural transformations. Also, from a software engineering perspective, it is clearly advantageous to keep the two sub-processes independent of each other. However, the extent to which this separation can be maintained is not evident. In certain cases, e.g. when there is no simple way to preserve the part of speech in translation, lexicon-driven modifications of the phrase structure could facilitate the process. In such cases, which are fortunately not very common, we will consider attachment of "transformation triggers" to nodes, based on the bilingual lexicon processing. Technically, this can be easily done in our specification formalism. The lexical phase itself will not make any changes in phrase structure, and only one pass of structural transformations will still be required.

Indeed, as [Melby 86] points out, the whole issue of lexical transfer and in particular the effect lexical transfer has on the overall quality of translation deserves further study.

3. Sense Distinction Characteristics

The highest level of distinction in the bilingual lexicon is the part of speech. Within each part of speech one may distinguish different senses (and specify lexical rules) as required in light of TL sense distinctions. According to one initially plausible view, it is possible to make use of the subcategorization information present in standard monolingual dictionaries to obtain the necessary syntactic information for disambiguating different senses of words in the SL. In fact, we have found this view to be untenable. Thus, for example, among the 13 senses that [Longman 78] distinguishes for the verb "hold", many are mapped to the Hebrew verb "החזיק", even when they belong to different syntactic categories as defined in Longman's system. On the other hand, consider the different uses of "hold" in the following sample sentences:

- She is holding the baby
- We held our breath in fear
- We were holding a meeting

Although Longman makes a distinction between the three senses (cases 1,3, and 13, respectively), all of them are classified as T1 verbs according to Longman's coding system, which is the only feasible instrument for computer reference. In Hebrew, different verbs must be used ("החזיק", "עצר", "קיים", respectively). Moreover, even under the same sense (10), two sample sentences are given, which in Hebrew require different verbs:

- What he said still holds ("תקף")
- Can the good weather hold? ("המשיך")

As a second example of the inadequacy of standard monolingual subcategorization, consider Longman's class of T3 verbs - verbs which take infintival complements with "to" and NP objects. E.g.:

- want Ron to win
- ask Ron to leave
- tell Ron to come

However, each of the corresponding Hebrew verbs has a distinct subcategorization frame:

- "רוצה שרון ינצח" (want that Ron will win)
- "מבקש מרון לעזוב" (ask from Ron to leave)
- "אומר לרון לבוא" (tell to Ron to come)

Note that the form

- "מבקש שרון יעזוב" (ask that Ron will leave)

is also valid in Hebrew, but as a translation of "ask that Ron leaves" rather than "ask Ron to leave". Therefore, Transfer and not Generation seems to us the natural place to decide on the appropriate form.

These examples illustrate the lack of isomorphism between subcategorization patterns for specific verbs in English and their counterparts in Hebrew. It is reasonable to expect that, generally, subcategorization is not invariant under translation between most SL-TL pairs (cf e.g. [Warwick 87]). As [Nagao 86] points out, "it is not exceptional, but rather usual, that a verb of SL has to be translated into different target lexical items, even though the native speakers of SL cannot clearly realize the meaning difference". Consequently, we have found it useful to construct algorithms which map lexical items, considering subcategorization properties and sense distinctions of both SL and TL. This is especially useful for verbs, but the same specification formalism is used for all other lexical categories, making use of syntactic information in different ways.

We anticipate that the construction of a large scale lexicon will be facilitated by the existence in the SL of subclasses of items, in each lexical category, which have identical or highly similar subcategorization frames, and which correspond to items in the TL with similar frames. The entries for the elements of such subclasses can be handled by algorithms whose statements have more or less the same selection conditions.

In constructing the BL lexicon of our system we have followed a lexicalist view of syntax. In particular, our view of the interaction of the syntactic component and the lexicon in Transfer is inspired by the projection principle [Chomsky 1981]. This principle states that the syntactic structure of a phrase (at any level of representation) is a projection of the argument structure imposed by the lexical entry of the head of the phrase. Mapping of a lexical item in the SL onto a counterpart in the TL depends upon a matching of the subcategorization frames of the two items. This matching requires recognition of an SL item in a tree as the head of a sub-phrase which satisfies the argument structure specified by the antecedent conditions of one of the statements in its lexical entry. Translation of any sub-clause begins with its head, as this determines the argument structure of the clause.

It is necessary to qualify this characterization of our approach in an important respect. Although our algorithms are stated primarily in terms of the syntactic information provided by the parser, we have found it necessary to supplement this information with a restricted list of general semantic features. For example, consider the verb "run". On its intransitive use, it translates into Hebrew as "רץ", while its transitive use corresponds to the Hebrew causative verb "הפעיל". However, intransitive "run" allows certain NP adverb complements, as in "run a mile". To identify these NPs as adverbial phrases and so preserve the intransitive sense of "run", we attach attributes Time or Distance to the entries of nouns like "time" and "mile" in the SL lexicon. The algorithm for "run" can then recognize it as having its intransitive sense when it takes only one complement and the head N of this NP contains one of these attributes. Features such as Human/Non-human, Concrete/Abstract, etc., and a small number of domain specific features (e.g. hardware device, software component, state, etc., for the domain of computer manuals) are also included.

There is yet another kind of feature marker that we include in the SL lexicon, and which we call "list marker". These markers assist disambiguation in cases where a few nouns can be grouped together, usually for computational efficiency, although their common denominator has no obvious name. For example, in the sense distinction for the verb "assume", if the complement is an NP with a noun in the set [office, chairmanship, position,...], then the Hebrew verb-equivalent is different from other sub-cases of "assume". We give this group of nouns a name and mark the nouns in the group accordingly. Then one can refer to this name as a feature. Indeed, nouns in such groups have semantic similarities, and in principle, they are equivalent to traditional semantic markers. The point is that list markers can be formed in an ad-hoc fashion, without worrying about the generality of the group, or finding an appropriate label for it.

The different mechanisms presented above provide, in fact, various levels of characterization of the relation between a given verb and a noun (or an NP) to which it refers: subcategorization requirement for the very existence of a noun (NP) in a given position (e.g. as a direct object); a more constricting requirement for the existence of a noun with specified semantic features (defined as such, or by means of lists, in the SL lexicon); and a particular requirement for the existence of a specific noun (or group of nouns).

The sense distinction for multi-sense nouns (homonyms and polysemes) and other parts of speech is done according to basically the same strategy which we use for verbs; namely application of context-sensitive differentiating rules. Generally, nouns and adjectives are less ambiguous for translation, but the ambiguity, when it exists, is more difficult to resolve at the sentence level without extensive semantic and pragmatic knowledge. Still, many cases can be resolved by rules of the kind shown above. For example, although nouns, unlike most verbs, do not require complement structures, disambiguation in the source language can sometimes be facilitated by rules which refer to the presence of optional N complements. Consider, for example, the noun "statement". When it has a 'that' S' complement ("a statement that Rina has been promoted") it translates to Hebrew "הודעה" when it occurs with a PP complement headed by "of" with an NP object headed by an inanimate noun ("statement of the theory"), it translates to "ניסוח".

When sense distinction cannot be expressed in terms of our system (e.g. for the homonymous noun "bank"), we specify only the more common sense in the given text domain. In the future, we may mark such cases so that the information about the lexical ambiguity can be made available to an interactive disambiguation module.

For adjectives, the characterization is done in most cases by reference to semantic features of the noun(s) they modify. In some cases different translations will be required for an adjective when it appears as a prenominal modifier or in the VP predicative form:

- My old friend is not old
  "החבר הותיק שלי אינו זקן"

The syntactic distinction between the two forms of "old", and the selection of the different Hebrew equivalents ("זקן", "ותיק"), are easy to specify in our formalism (cf. the definition of N-of-ADJ in section 6 below).

4. Some Methodological Aspects

From the discussion and examples above, it is apparent that our lexicon specifies bilingual information in great detail. This may seem to conflict with certain modern approaches to Transfer methodology, where the guiding view is "small (and simple) is beautiful". [Isabelle 86], [Arnold 87], and others, advocate an approach where the BL lexicon states only facts like:

know -> wissen (in German; or "savoir" in French;
or "לדעת" in Hebrew,
when the verb takes a sentential object)
know -> kennen (in German; or "conaitre" in French;
or "להכיר" in Hebrew,
when the verb takes a nominal object)

The selection of the correct translation is then done in the Generation phase, based on restrictions on the target language. We claim that this approach is problematic for the following reasons.

First, the "know" example, although widely quoted in the literature, is rather simplistic. Even in the monolingual subcategorization for "know" in [Longman 78], there are 15 different frames, many of which require a different verb, or verb-form, use of a preposition, or even a completely different syntactic structure in Hebrew. In addition, there are verb-particle and other collocations which would be quite difficult (and unnatural) to handle if decisions were postponed to the monolingual Generation ("know of", "know better", "know X to be Y", are some examples which hold not only for Hebrew). It is not single words but patterns and structures that must be handled simultaneously.

Second, one should note that the mechanism presented in this paper may also, in principle, be applied to Generation rather than Transfer; the conditions will then be stated in terms of the target language. By the same token, certain differentiating rules may be also applied as part of Analysis (or post-Analysis). However, we feel that the right place for this kind of processing is in Transfer. At least when the target and source languages are linguistically remote (as, for example, in the case of English and Hebrew), severely restricting the scope of BL operations may result in loss of information vital for translation on the way to Generation; e.g. dependence on tense or voice, when the structural expressions of these properties are different in the SL and TL. Alternatively, such information could be carried forward to Generation, but then Generation loses its primarily monolingual nature.

Allowing Transfer to pass forward alternative translations for SL lexical entries, may result in a large number of possible sentence translations which must be handled by Generation. Even if conceptually viable, this strategy is computationally highly unattractive. A similar comment is often made in other contexts (e.g. Analysis - cf. [Stallard 87]).

Finally, it should be emphasized that keeping Transfer small is not, in itself, the crucial issue. More important, it seems to us, is the isolation of bilingual considerations. (Isolation should not be mistaken for serialization of SL-BL-TL operations, which is not at all required. Ideally, isolated modules could even work in parallel, as suggested by [Isabelle 86]). In fact, trying to minimize bilingual Transfer at any cost may yield unnecessarily complex Generation and/or Analysis, which are forced to handle problems that are not inherently within their scope. If, for example, Generation has to consider patterns that were only created by phenomena in a certain source language but were not fully resolved in Transfer, then minimizing Transfer represents no real gain in modularity or language independence. Nor does it save lexicographic effort, as in any case, the linguistic classification and judgement process remains basically the same, even if shifted to other system components.

A remark is in order concerning multi-target generality. In principle, it is technically possible to add tags at the *put* statement, along with the HEB tag, to define multilingual translations (cf. e.g. Boitet 86]). However, from our discussion of the difference between monolingual subcategorization patterns and the sense distinctions needed for translation, it follows that the algorithms in the lexicon must be further refined in order to cover all sub-senses needed for a set of given target languages. For any one language in the set, many of the differentiating rules will be redundant (see "voir trad n" definitions in [Boitet 86]). In practice, it would be better to construct distinct variants of the lexicon for different target languages. This need not significantly increase the lexicographic work required, as it is possible to use an existing lexicon as the basis for constructing new variants. The linguistic re-evaluation of senses is required anyway. What is important is that all target languages use the same structure and formalism.

5. The Specification Language

[Slocum 87] claims (correctly, in our view) that "lexical entries for computer use tend to be formally stated, compact, and thus cryptically encoded". While the formal style is inevitable, we have tried to avoid compactness and cryptical expression by allowing the lexicographer to state the lexical facts and the effects they have on processing in terms that are directly related to the logic of the linguistic process. Therefore, the set of available instructions is rather simple and intuitive. We have tried to allow enough expressive power to support a variety of requirements for bilingual lexical mapping, while restricting the scope of operations as much as possible, to reduce complexity and avoid undesired side effects on other entries or subsystems. We have also emphasized ease of maintenance and testing, and strict isolation of the lexical subsystem from other parts of the translation system.

Each entry in the bilingual lexicon is in fact a mini-program. Although executable declarations may look complicated at first glance, they have, in our view, many advantages over a rigid data structure. The specification formalism may be less neat and "natural" than the metalanguage [Isabelle 86] hopes for, but it can be made more "user friendly" through the introduction of higher level abbreviations on top of the basic language, as required by users. Some functions for abbreviated writing were mentioned in section 2 above. Other mechanisms (e.g. macros) can be used to tailor the specification style to individual tastes.

The program which comprises a lexical entry is initiated by reference to a lexical item that appears on the SL parse tree. The program terminates when an *end* instruction is reached, or when the last instruction in the program is executed. A function is terminated by a return instruction.

The instructions allow the lexicographer to check for the existence of a particular condition or pattern on the parse tree by an appropriate *if* instruction; control the sequencing of instructions by using a *goto* instruction; and decide on a particular translation for a word by using a *put* instruction (this is not allowed in the case of a function).

The following represents the current inventory of lexical operations supported by the specification formalism. It may be necessary to extend this formalism as more experience is gained with actual lexical entries.

The syntax of the specification language is given below. Bold letters denote non-terminal constructs, normal letters denote terminals (keywords). The notation used is as follows:

symbol	description	symbol	description
= =	a meta-symbol.	[term]	one or no appearance of term.
|	a meta-symbol denoting choice.	(term)$^+$	one or more appearances of term.
nil	a representation of the null string or list.	(term)*	zero or more appearances of term.
( ) < > ;	terminals.		

program = = start **access-key statement** $^+$ finish

statement = = [**label** :] **instruction**

instruction = = end; | goto(**label**); | put [in **n-name**] (**allowed-attribute value**); | call **function-name(parameter-list)**; |
if (condition) then < **instruction** $^+$ > [else < **instruction** $^+$ >];

function-name = = **user-defined-function-name** | **pre-defined-function-name**

user-defined-function = = define **user-defined-function-name fn-statement** $^+$ finish

fn-statement = = [**label** :] **fn-instruction**

fn-instruction = = goto(**label**); | return(true); | return(false); | call **pre-defined-function-name(parameter-list)**; |
if (condition) then < **fn-instructio n** $^+$ > [else < **fn-instruction** $^+$ >];

condition = = call **function-name** | **simple-pattern** | not (**condition**) | (**condition**) and (**condition**) | (**condition**) or (**condition**)

simple-pattern = = **n-name(attribute-pair** $^+$ **)** | **n-name** has [no] **relation pattern**

pattern-condition = = **node** has [no] **relation pattern**

node = = [**n-name**](**attribute-pair** $^+$) | **n-name**

relation = = [first] prebrother | [first] postbrother | [first] son | father | ancestor |
[last] prebrother | [last] postbrother | [last] son | descendent | relative

pattern = = **simple-pattern-condition** | (**pattern-condition**) | (**pattern** [not] before **pattern**) | (**pattern** [not] after **pattern**)

simple-pattern-condition = = **node** | (**node**,) $^+$ **node** | (**node** or **node**) | (**node** and **node**)

n-name = = main | cn | **name**

parameter-list = = nil | **parameter** $^+$

attribute-pair = = (**attribute-name value** $^+$)

value = = string | key | (**value** $^+$)

label = = **name**

access-key = = lexical item followed by part of speech.

attribute-name = = **name**

allowed-attribute = = an attribute-name the user is allowed by the system to change.

pre-defined-function-name = = a name of a function defined by the system.

user-defined-function-name = = name

name = = a string of letters and digits starting with a letter.

parameter = = a parameter passed to a function.

comment = = each line starting with @ is a comment.

The programs and functions in the dictionary include names for various entities. The scope of these names is limited as much as possible in order to facilitate debugging and maintenance (e.g. a name of a label or of a particular pattern (node) is recognized only within the program in which it appears). The user's ability to define variables and names is also intentionally restricted. There are two special names: MAIN, the name of the root of the parse tree; and CN (Current Node), the name of the leaf node representing the word for which the program was invoked.

Additional information about the semantics of names, a formal definition of the relations between nodes (e.g. prebrother, first prebrother, descendent, relative, etc.) and the keywords Before and After, etc., are provided in [Golan 88], an extended version of this paper.

6. An Additional Example

The following is the function N-of-ADJ, introduced in section 2 above, which identifies the head noun argument, with property P, of a given adjective. It is a somewhat simplified version which deals with the major relevant phrase structures. It is given here mainly to illustrate the expressive power of the specification formalism as a general tool for pattern matching on parse trees.

```
Define N-of-ADJ
@ The function N-of-ADJ returns a True/False value
@ The parameter P refers to a feature that can get + /- values

1 If (CN has Ancestor ( A ((CAT AJP)) has Postbrother
        F((CAT NP NOUN)(P +))))
    Then < Return(True); > ;

   @  John is an old and valued friend.
   @  John spoke to many old and valued friends.

2 If (CN has Ancestor (A((CAT AJP)) has Prebrother
        F((CAT NP NOUN PRON) (WH nil)(P +))))
    Then < Return(True); > ;

   @  John is very old.
   @  Mary is very old and tired.

3 If (CN has Ancestor ( A ((CAT QUES)) has First Son
        F ((CAT NP PRON)(WH 1))))
    Then < If (F ((P +))) Then < Return(True); >
    Else < Return(False); > ; > ;

   @  Who is competent?
   @  Who does John expect to seem old?
   @  What does John expect to seem old?

4 If (CN has Ancestor B((CAT INFCL)))
    Then < If(((B has Prebrother
          F ((CAT NP NOUN PRON)(WH nil)(P +)))
        or (B has Father ( C ((CAT AJP)) has Prebrother
          F ((CAT NP NOUN PRON)(WH nil)(P +)) ) ) )
        or (B has Father ( C ((CAT INFCL)) has Prebrother
          F ((CAT NP NOUN PRON)(WH nil)(P +)) ) ) )
        Then < Return(True); > ; > ;

   @  John wants to be rich and famous.
   @  Bill is certain to be happy.
   @  Mary wants to seem to be competent and punctual.

5 If ( CN has Ancestor ( A ((CAT RELCL)) has Prebrother
        F((CAT NOUN)(P +)))) Then < Return(True); > ;

Finish
```

References

[**Arnold 87**] Doug Arnold and Louis des Tombe: "Basic Theory and Methodology in Eurotra", in *Machine Translation: Theoretical and Methodological Issues*, Sergei Nirenburg (ed.), Cambridge University Press, 1987.

[**Biewer 85**] Axel Biewer, Christian Feneyrol, Johannes Ritzke and Ervin Stegentritt: "ASCOF - A Modular Multilevel System for French - German Translation", *Computational Linguistics, vol 11*, no. 2-3, April-September 1985.

[**Boitet 86**] Christian Boitet and N. Nedobejkine: "Toward Integrated Dictionaries for M(a)T", Proc. of COLING'86, Bonn, August 1986.

[**Chomsky 81**] Noam Chomsky: *Lectures on Government and Binding*, Foris Publications, Dordrecht, Holland, 1981.

[**Cullingford 87**] Richard E. Cullingford and Boyan A. Onyshkevych: "An Experimentation in Lexical-driven Machine Translation", in *Machine Translation: Theoretical and Methodological Issues*, Sergei Nirenburg (ed.), Cambridge University Press, 1987.

[**Golan 88**] Igal Golan, Shalom Lappin and Mori Rimon: "An Active Bilingual Lexicon for Machine Translation", Technical Report 88.242, IBM Israel Science and Technology, July 1988.

[**Isabelle 86**] Pierre Isabelle and Elliott Macklovitch: "Transfer and MT Modularity", Proc. of COLING'86, Bonn, August 1986.

[**Jensen 86**] Karen Jensen: PEG 1986: A Broad-coverage Computational Syntax of English, Technical Report, IBM T. J. Watson Research Center, February 1986.

[**Longman 78**] *Longman Dictionary of Contemporary English*, Longman Group Ltd., Harlow and London, England, 1978 (1986 edition).

[**Melby 86**] Alan K. Melby: "Lexical Transfer: a Missing Element in Linguistic Theories", Proc. of COLING'86, Bonn, August 1986.

[**Nagao 86**] Makato Nagao and Jun-ichi Tsujii: "The Transfer Phase of the Mu Machine Translation System", Proc. of COLING'86, Bonn, August 1986.

[**Slocum 87**] Jonathan Slocum and Martha Morgan: "The Role of Dictionaries and Machine-Readable Lexicons in Translation", International Lexicon Workshop, Stanford, July-August 1987.

[**Stallard 87**] David Stallard: "The Logical Analysis of Lexical Ambiguity", 25th Annual Meeting of the ACL, Stanford, July 1987

[**Warwick 87**] Susan Warwick: "Automated Lexical Resources in Europe", International Lexicon Workshop, Stanford, July-August 1987.