MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1166 August, 1989

CONCEPTUAL BASIS OF THE LEXICON IN MACHINE TRANSLATION

Bonnie J. Dorr

Abstract: This report describes the organization and content of lexical information required for the task of machine translation. In particular, the lexical-conceptual basis for UNITRAN, an implemented machine translation system, will be described. UNITRAN uses an underlying form called *lexical conceptual structure* to perform two difficult, but crucial, tasks: *lexical selection* (*i.e.*, choosing the appropriate target-language terms for a given source-language sentence) and *syntactic realization* (*i.e.*, mapping an underlying lexical representation to a corresponding syntactic structure). Lexical word entries have two levels of description: the first is an underlying lexical-semantic representation that is derived from hierarchically organized primitives, and the second is a mapping from this representation to a corresponding syntactic structure. The interaction of these two levels will be discussed and the lexical selection and syntactic realization processes will be described.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research has been provided by a grant from the Kapor Family Foundation and by NSF Grant DCR-85552543 under a Presidential Young Investigator's Award to Professor Robert C. Berwick. Useful guidance and commentary during this research were provided by Bob Berwick, Michael Brent, Bruce Dawson, Sandiway Fong, and Mike Kashket. This report is a revised version of a paper appearing in the Proceedings of the First Annual Workshop on Lexical Acquisition, IJCAI-89, Detroit, MI.

1 Introduction

The work of Jackendoff [1983] has been the primary influence on the design of the lexical-semantic component of UNITRAN, a machine translation system that translates Spanish, English, and German bidirectionally.[1] The representation adopted is *lexical conceptual structure* (henceforth LCS) as formulated by Hale & Laughren [1983] and Hale & Keyser [1986]. Consider the following translation from English to Spanish:

(1) I stabbed John ⇒ Yo di cuchilladas a Juan
(I gave knife-wounds to John)

In this example, the source language sentence diverges both syntactically and lexically from the target language sentence. The syntactic divergence shows up in the realization of the verbal object as a noun phrase *John* in English and as the prepositional phrase *a Juan* (literally, *to John*) in Spanish. The lexical divergence shows up in the realization of the main action as the single verb *stab* in English and as the composite form *dar cuchilladas* (literally, *to knife* or *to give knife-wounds*) in Spanish. The UNITRAN system overcomes these types of divergences by providing a framework within which lexical-semantic information is abstracted to a level that is distinct from that of syntactic information; target-language terms are then selected and realized on the basis of an interaction between these two levels.

The LCS that describes the *stab* event is:

```
(2)  (CAUSE X
        (GO-POSS KNIFE-WOUND (TOWARD-POSS (AT-POSS KNIFE-WOUND Z)))
        (WITH-INSTR *HEAD* SHARP-OBJECT))
```

This LCS description provides the meaning "THING X causes a possessional transfer of a knife-wound to THING Z using a sharp object as an instrument." Section 2 will describe the LCS representation in more detail, and section 3 will introduce the mapping between this level of representation and the syntactic level. Section 4 presents the three processing modules of the LCS component: the LCS selection module, the LCS composition module, and the root word selection/realization module. We will concentrate primarily on the third module in section 5, describing how it applies to example (1). Throughout this report, we will see that the compositional nature of the LCS representation and the abstraction of lexical-semantics from syntax are crucial to the model presented here.

[1]This report discusses the lexical-semantic component of UNITRAN; for a description of the syntactic component, see [Dorr, 1987].

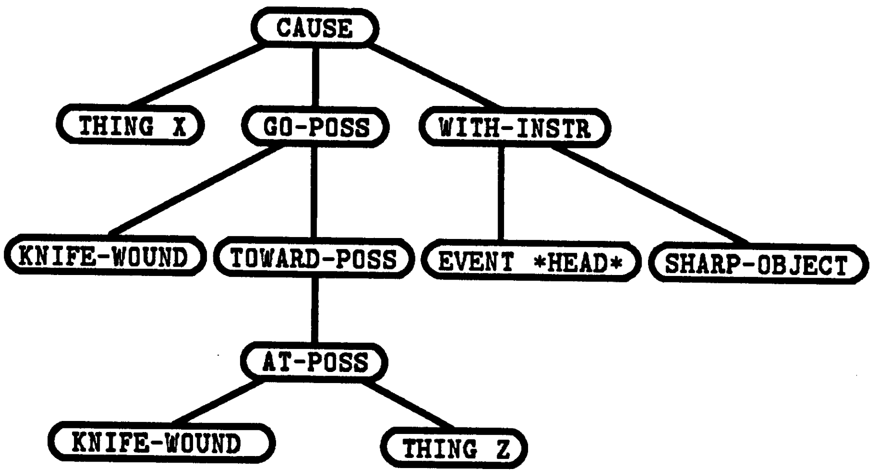


Figure 1: Underlying LCS for the Stab Event

2 The LCS Representation

UNITRAN requires a dictionary of lexical root word entries for each language that is processed by the system. Each entry has two levels of description: the first is a lexical-semantic representation (the LCS of the lexical word), and the second is a mapping from the LCS representation to the syntactic structure (category and structural positioning of each argument associated with the lexical word). This section presents the LCS representation, and the next section describes the mapping from the LCS to the syntactic structure.

Figure 1 shows the underlying LCS tree structure generated from (2). Note that the **SHARP-OBJECT** instrument argument is included in the LCS even though the source-language does not realize this argument in (1). Including this argument in the LCS allows flexibility in generating the target-language sentence, which may or may not require this argument to be realized. Thus, it would be possible to generate either *he stabbed the robber*, or *he stabbed the robber with a knife* (*scissors*, *poker*, *etc.*). The ***HEAD*** symbol is a place-holder that points to the overall *stab* event (*i.e.*, the *stab* event is performed with a sharp object).

Two properties of the LCS representation enable it to make the appropriate lexical selection and syntactic realization decisions for cases such as (1). The first is that the LCS representation is *compositional* in nature. Thus, the LCS representation underlying the *stab* event allows the overtly compositional form *dar cuchilladas* (literally, *to knife* or *to give knife-wounds*) to be selected as the target-language form for the inherently compositional verb *stab*. The second property of the representation is that it provides an abstraction of lexical-semantic information from syntactic information. Thus, the phrase *a*

LCS Type	*LCS Name*
EVENT	CAUSE, LET, GO-POSS, GO-IDENT, GO-TEMP, GO-LOC, STAY-POSS, STAY-TEMP
STATE	BE-IDENT, BE-POSS, BE-LOC, BE-TEMP
THING	BOOK, PERSON, REFERENT, KNIFE-WOUND
PROPERTY	TIRED, HUNGRY
PATH	TO-POSS, TO-LOC, FROM-POSS, FROM-LOC, TOWARD-POSS, TOWARD-LOC
POSITION	AT-POSS, AT-LOC, IN-LOC, ON-LOC, WITH-INSTR
LOCATION	HERE, THERE
TIME	TODAY, 9:00
MANNER	FORCEFULLY, LIKINGLY
INTENSIFIER	VERY

Figure 2: LCS Types and Names

Juan (literally, *to John*) is realized as the equivalent target-language form for *John*, even though there is no direct syntactic correspondence.

Figure 2 gives some examples of the lexical primitives used by the system. (Not all of the lexical primitives are listed here; see Dorr [forthcoming].) In particular, I adopt Jackendoff's notions of **EVENT** and **STATE**; these are further specialized into such primitives as **CAUSE**, **GO**, **BE**, **STAY**, and **LET**. The specialized primitives are placed into Temporal, Locational, Possessional, Identificational, Circumstantial, Instrumental, Intentional, and Existential fields. For example, the primitive **GO-POSS** refers to a **GO** event in the Possessional field (*e.g.*, Beth received (= **GO-POSS**) the doll). If the **GO** event were placed in a Temporal field, it would become **GO-TEMP** (*e.g.*, the meeting went (= **GO-TEMP**) from 2:00 to 4:00). One difference between Jackendoff's representation and the one shown here is that the two-place predicate **POSITION** (*e.g.*, **AT-POSS**, **AT-LOC**, **WITH-INSTR**, *etc.*) is used instead of the one-place predicate **PLACE**; thus, the **KNIFE-WOUND** argument in figure 1 appears both internally and externally to the **AT-POSS** LCS node. Although the system uses only a small set of lexical-semantic primitives (approximately 25), this set is quite adequate for defining a potentially large corpus of words due to the compositional nature of LCS's. The advantage of a small set of primitives is that the search space is reduced during the lexical-selection stage of generation. The importance of a small search space will become more apparent later when we look at the lexical selection process during a translation example in section 5.

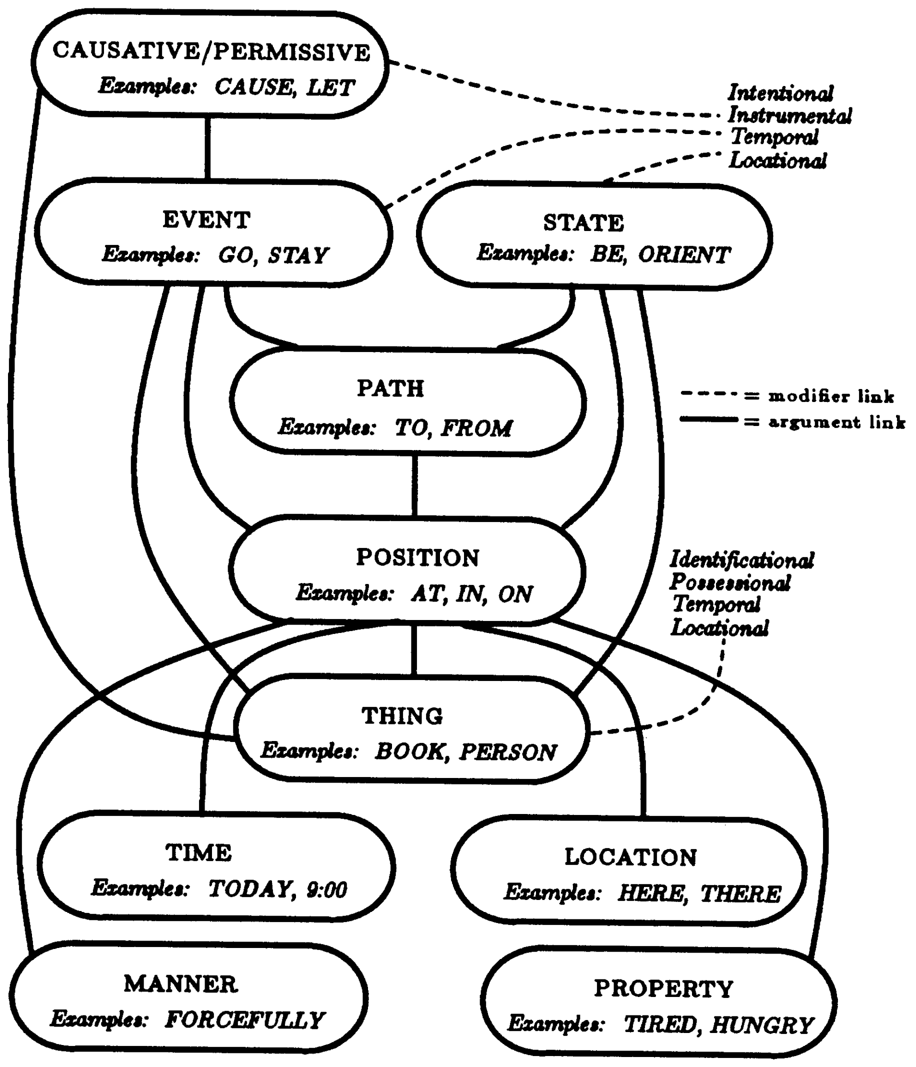


Figure 3: The LCS Hierarchy

The LCS representation associated with a given lexical entry is derived from the LCS hierarchy shown in figure 3 (some of the details have been omitted for simplicity). This hierarchy specifies the arguments (solid lines) and modifiers (dotted lines) that are allowed for each LCS represented in the system. The hierarchy is a graphical representation consistent with the well-formedness rules that are proposed in [Jackendoff, 1983] as shown in figure 4. The rules shown here have been adapted to the LCS scheme used in UNITRAN. In particular, Jackendoff does not provide a mechanism for handling modifiers

CAUSATIVE	$\rightarrow$	CAUSE (THING|EVENT, EVENT)
Modifier Fields: Intentional, Instrumental, Temporal, Locational		
PERMISSIVE	$\rightarrow$	LET (THING|EVENT, EVENT)
Modifier Fields: Intentional, Instrumental, Temporal, Locational		
EVENT	$\rightarrow$	f_{event}(THING, PATH|POSITION)
Modifier Fields: Intentional, Instrumental, Temporal, Locational		
STATE	$\rightarrow$	f_{state}(THING, PATH|POSITION)
Modifier Fields: Intentional, Instrumental, Temporal, Locational		
PATH	$\rightarrow$	f_{path}(POSITION)
POSITION	$\rightarrow$	
$f_{position}$(THING|EVENT, THING|TIME|PROPERTY|LOCATION|MANNER)		
THING	$\rightarrow$	*lexical item*
Modifier Fields: Identificational, Possessional, Temporal, Locational		
TIME	$\rightarrow$	*lexical item*
PROPERTY	$\rightarrow$	*lexical item*
LOCATION	$\rightarrow$	*lexical item*
MANNER	$\rightarrow$	*lexical item*

Figure 4: Well-Formedness Rules Represented in the LCS Hierarchy

(*i.e.*, non-argument positions). In the UNITRAN adaptation, modifiers (which are associated with **EVENTS**, **STATES**, and **THINGS**) are specified as LCS fields instead of LCS types. The use of fields provides a more flexible means of specifying LCS modifiers. For example, the noun *book* may be modified by a number of different types in the possessional field including **POSITION** (*the book for John*), **STATE** (*John's book*), **EVENT** (*the book that John owns*), *etc.*

Lexical entries are organized into LCS classes that are built on each LCS derived from the hierarchy. For example, the *stab* entry is in a class that is built on the GO-POSS LCS shown in figure 5. Thus, it inherits the arguments Y and Z, and, because it is an **EVENT**, it optionally takes modifiers in the Intentional, Instrumental, Temporal, and Locational fields as specified in the LCS hierarchy.

In addition to specifying the appropriate LCS representation, lexical classes provide CAUSATIVE and PERMISSIVE information (if any). For example, the word *fall* has a CAUSATIVE form *throw (down)* and a PERMISSIVE form *drop* as shown in figure 6. Figure 7 shows some of the lexical entries associated with the GO-POSS LCS of figure 5. (The variable U refers to a locally introduced modifier.)

Note that the CAUSATIVE and PERMISSIVE forms specify information about the logical subject. In the case of *give*, the subject is not coreferen-

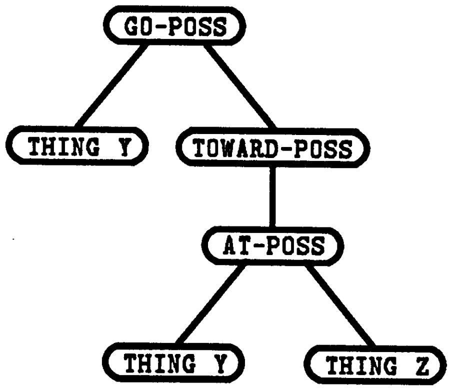


Figure 5: The GO-POSS LCS Associated with the Word Class that Contains the *stab* Entry

(GO-LOC X (TO-LOC (AT-LOC X Y)))	
fall	(Y = DOWN)
CAUSATIVE:	
throw	(SUBJECT = NIL); (Y = DOWN)
PERMISSIVE:	
drop	(SUBJECT = NIL); (Y = DOWN)

Figure 6: The LCS Class that Contains the *fall*, *throw (down)*, and *drop* Entries

(GO-POSS Y (TOWARD-POSS (AT-POSS Y Z)))	
go	
CAUSATIVE:	
give	(SUBJECT = NIL)
repossess	(SUBJECT = Z)
stab	(SUBJECT = NIL); (Y = KNIFE-WOUND); (MODIFIER = U), INSTRUMENTAL (WITH) SHARP-OBJECT
cut	(SUBJECT = NIL or U); (Y = KNIFE-WOUND); (MODIFIER = U), INSTRUMENTAL (WITH) SHARP-OBJECT

Figure 7: The Lexical Entries in the GO-POSS LCS Class

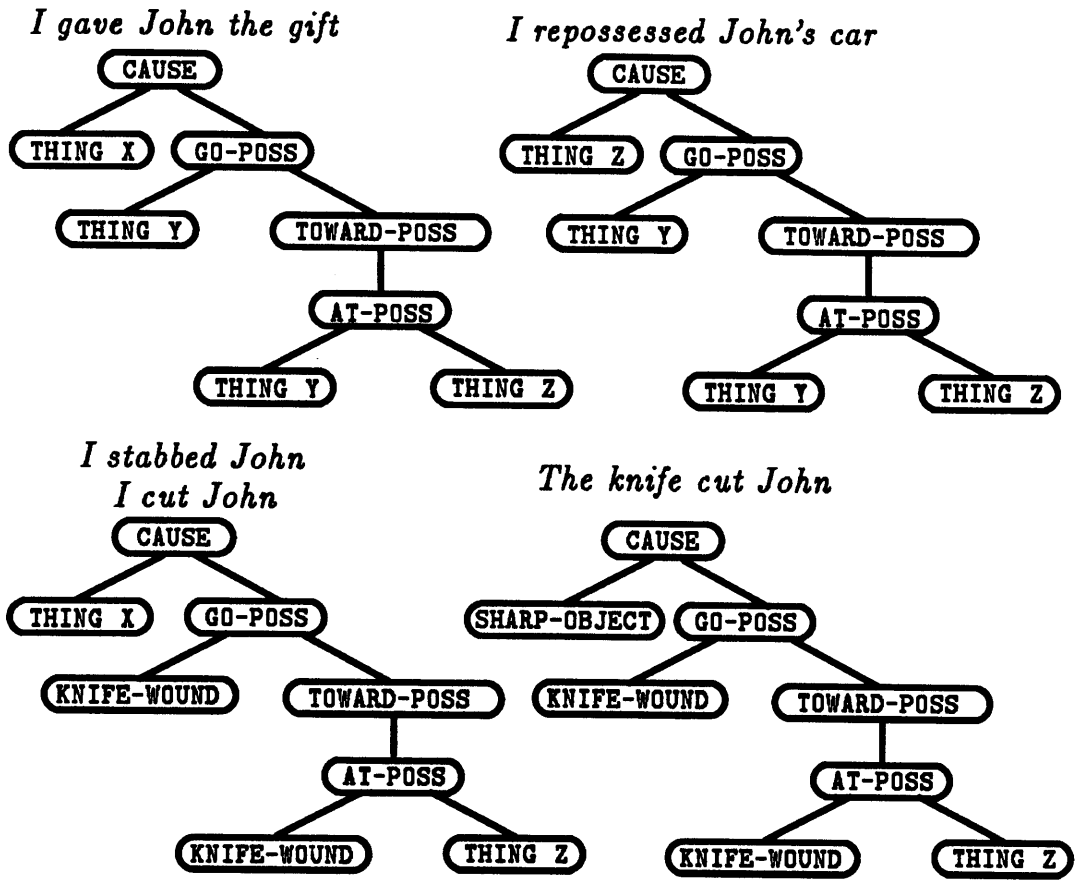


Figure 8: Causative Trees Represented by Definitions in the GO-POSS LCS Class

tial with any other arguments (*I gave John the gift*). In contrast, *repossess* requires the subject to be coreferential with the recipient Z since the subject of the event is also the recipient of the possessional transfer (*I repossessed John's car*). In the case of *stab* and *cut*, the subject is non-coreferential (*I stabbed/cut John*), or, in the case of *cut*, the subject may be coreferential with an instrument (*the knife cut John*). The causative trees represented by these definitions are shown in figure 8.

The LCS hierarchy and its associated LCS classes offer a number of benefits. First, arguments and modifiers need not be stated (unless they are required for coreference information) because lexical entries automatically inherit argument and modifier information from the LCS hierarchy. Second, the hierarchy allows lexical items to share LCS information without requiring that each entry explicitly spell out the details of the LCS representation; this eliminates proliferation of redundancy across lexical entries. Third, the LCS classes allow CAUSATIVE and PERMISSIVE forms to specify their idiosyncratic argument information by means of coreference and other types of argument indexation. Thus, words with related meanings may be grouped together despite different argument specifications; this avoids having to define these words independently in unrelated lexical entries. Finally, the LCS classes allow syntactically distinct lexical items to be defined in the same class if they have the same word sense. For example, the nominalized verb *entrance* is defined in the same LCS class (GO-LOC) as its verbal counterpart *enter* even though these two words are not of the same syntactic category.

The next section describes the mapping from the LCS representation presented in this section to the target-language syntactic form generated by the UNITRAN system.

3 Mapping From LCS to Syntactic Structure

In order to select and realize the appropriate target-language structures for source-language sentences, lexical entries must specify certain language-specific syntactic information. This is the nature of the LCS-to-syntax mapping associated with the definition of a word. The LCS-to-syntax mapping is incorporated into a word definition by means of three mechanisms. The first mechanism consists of two markers, :INT and :EXT, that map an LCS argument structure to a predicate-argument structure. A predicate-argument structure is an explicit syntactic representation of hierarchical relations between a predicate and its arguments. In particular, a predicate-argument structure embodies the asymmetry between the external argument position (*i.e.*, the subject of a verb) and the internal argument positions (*i.e.*, the objects of a verb). According to Rappaport & Levin [1986], language-specific linking rules relate variables in an LCS to positions in a predicate-argument structure. For example, in English the *agent* argument is mapped to a position that is external to the predicate. In UNITRAN, this process is implemented as a single, more general, language-independent linking routine that maps the hierarchically highest argument in the LCS to a syntactically external position, and all other arguments to syntactically internal positions. When this routine is to be overridden by a lexical entry, the language-specific markers :INT and :EXT are used.

The second LCS-to-syntax mechanism is the :CAT marker that provides a syntactic category for an LCS argument. According to Chomsky [1986], there is a language-specific function called CSR (canonical-syntactic representation) that provides a default mapping from each thematic-role to its appropriate syntactic category. For example, in English the *agent* role is mapped to the category N. In UNITRAN, the CSR function has been implemented as a general

LCS Type	*Syntactic Category*
EVENT	V
STATE	V
THING	N
PROPERTY	A
PATH	P
POSITION	P
LOCATION	ADV
TIME	ADV
MANNER	ADV
INTENSIFIER	ADV

Figure 9: CSR Mapping from LCS Type to Syntactic Category

language-independent routine that maps an LCS type to a syntactic category (see figure 9). When this mapping is to be overridden by a lexical entry, the language-specific marker :CAT is used.

The third LCS-to-syntax mechanism, developed specifically for the UNITRAN system, is the '*' marker; this marker provides a pointer to the position where arguments are explicitly realized in the surface form. Figure 10 shows how the '*' notation is used for the English and Spanish lexical entries that correspond to the *stab* event. Note that the Spanish LCS for *dar* contains a **Z** argument that is realized at the level of **TOWARD-POSS**, whereas the corresponding argument in the English LCS for *stab* is realized at the lowest level. This accounts for the distinction between the noun phrase *John* and the prepositional phrase *a Juan* in the translation example (1). Also note that the English LCS for *stab* contains a **KNIFE-WOUND** argument corresponding to the **Y** argument in the Spanish LCS for *dar*. Since this argument is not associated with a '*' marker in the English entry, it is considered to be *conflated* or "understood" as part of the meaning of the verb *stab*. However, the corresponding **Y** argument in the Spanish entry is associated with a '*' marker, thus indicating that it must be explicitly realized for the verb *dar*. This corresponds to the distinction between *stab* and *dar cuchilladas* in (1). The ***EXTERNAL*** symbol used in the LCS definition of *a* is a place-holder for an LCS that will fill this position by means of lexical-semantic composition (to be described in the next section). For example, when the LCS associated with *a* is composed with the LCS associated with *dar*, the **Y** argument will replace the ***EXTERNAL*** place-holder in the LCS associated with *a*.

The '*' marker is required for every explicitly realized argument of a

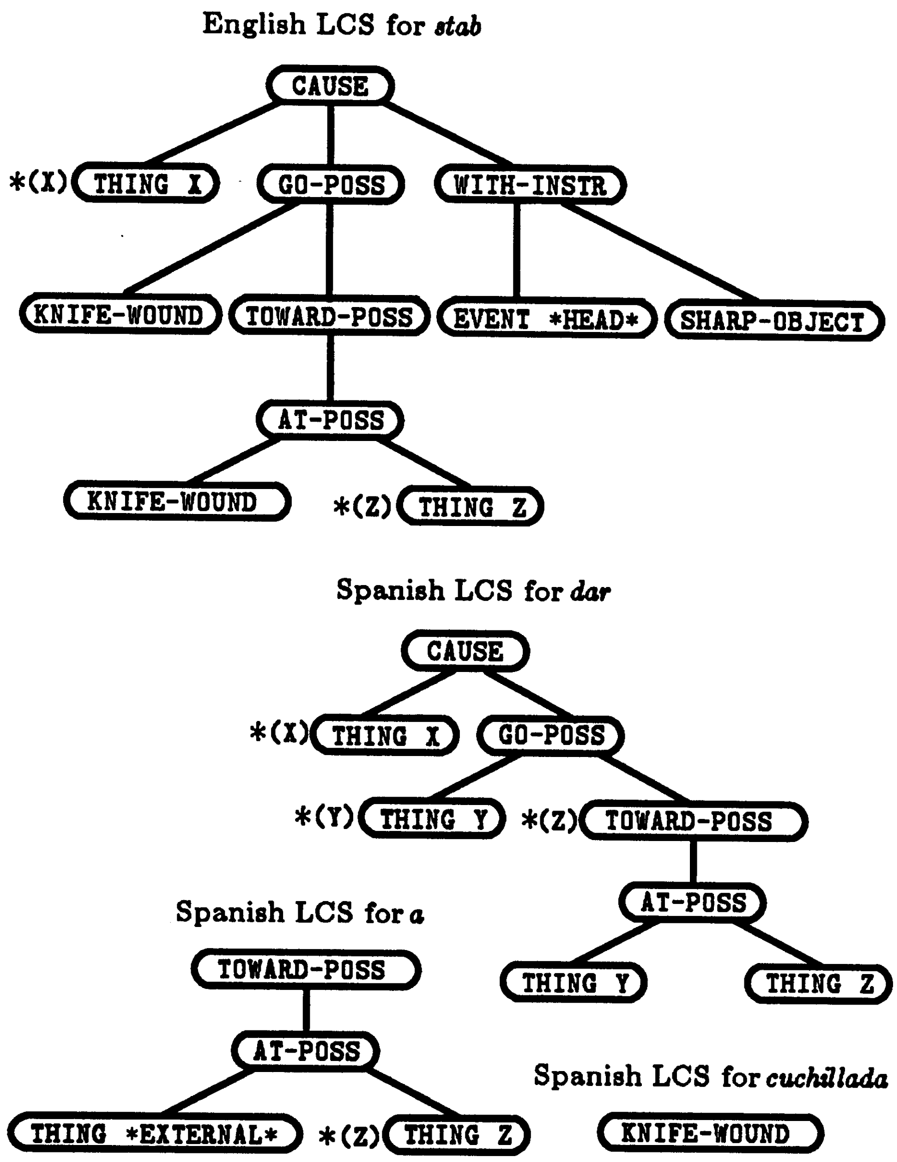
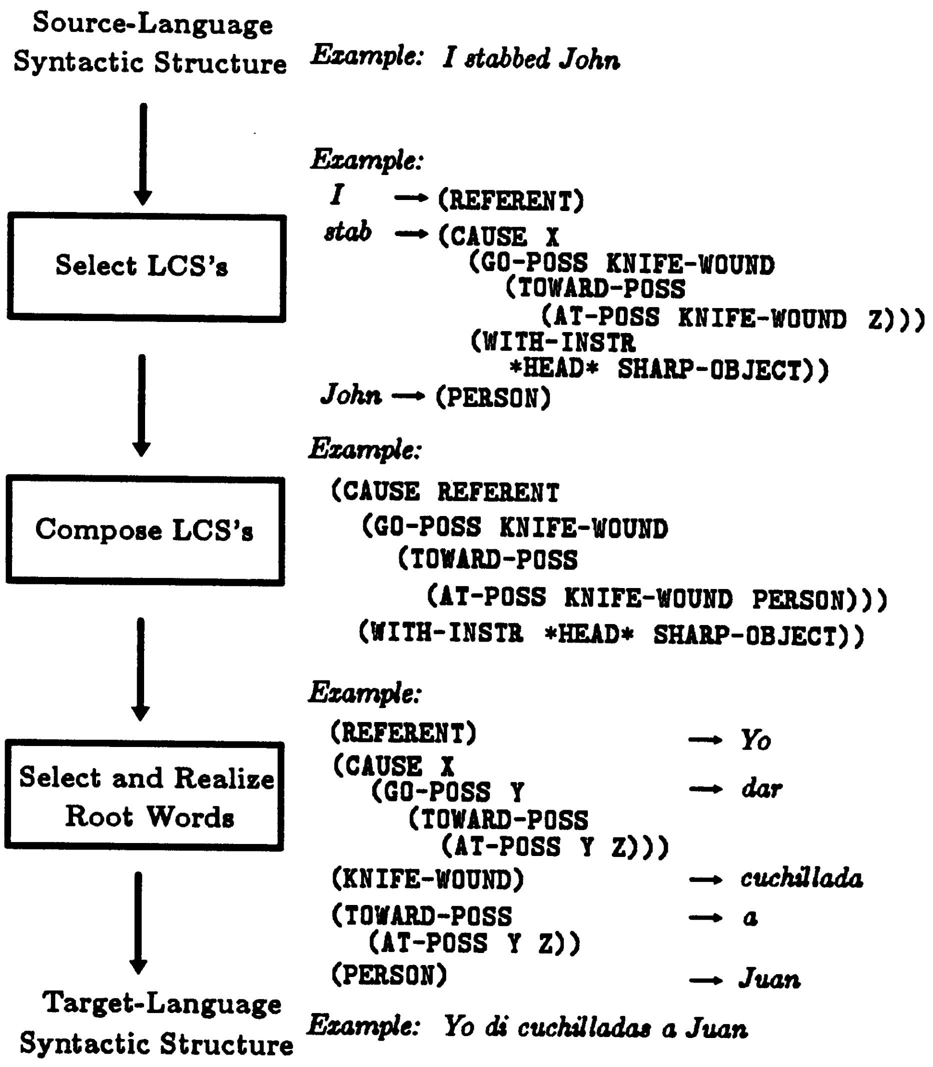


Figure 10: English and Spanish Lexical Entries for the *stab* Event


Source-Language Syntactic Structure
Select LCS's
Compose LCS's
Select and Realize Root Words
Target-Language Syntactic Structure

Example: I stabbed John

Example:

```
I     → (REFERENT)
stab  → (CAUSE X
          (GO-POSS KNIFE-WOUND
             (TOWARD-POSS
                (AT-POSS KNIFE-WOUND Z)))
          (WITH-INSTR
             *HEAD* SHARP-OBJECT))
John  → (PERSON)
```

Example:

```
(CAUSE REFERENT
   (GO-POSS KNIFE-WOUND
      (TOWARD-POSS
         (AT-POSS KNIFE-WOUND PERSON)))
   (WITH-INSTR *HEAD* SHARP-OBJECT))
```

Example:

```
(REFERENT)                     → Yo
(CAUSE X
   (GO-POSS Y                  → dar
      (TOWARD-POSS
         (AT-POSS Y Z)))
(KNIFE-WOUND)                  → cuchillada
(TOWARD-POSS                   → a
   (AT-POSS Y Z))
(PERSON)                       → Juan
```

Example: Yo di cuchilladas a Juan

Figure 11: Design of the LCS Component in UNITRAN

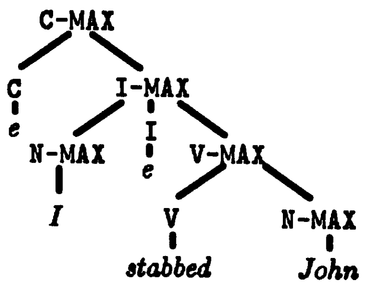


Figure 12: Source Language Syntactic Tree for *I stabbed John*

lexical entry, whereas the :INT, :EXT, and :CAT markers are used only in cases where the linking and CSR functions need to be overridden.

4 Processing Modules of the LCS Component

Figure 11 shows the design of the LCS component of the UNITRAN system as described in Dorr [forthcoming]. Essentially, the lexical-semantic component performs three top-level tasks. The first is the mapping of each word to its corresponding LCS. The second is the composition of the resulting LCS forms into a single LCS that underlies the source- and target-language sentences. The third is the mapping of each node in the composed LCS to an appropriate target-language word, which is then projected to its phrasal (or *maximal*) level and attached according to the positioning requirements of the word that selects it.[2]

We return to our translation example shown in (1), repeated here as (3):

(3) I stabbed John ⇒ Yo di cuchilladas a Juan
(I gave knife-wounds to John)

The parser in the syntactic component of UNITRAN supplies a source-language syntactic tree to the LCS component of the system. Figure 12 shows the source-language syntactic tree input for the current example.[3] When this syntactic tree is passed to the LCS component, the LCS's corresponding to the words *I*, *stab*, and *John* are selected, and a single underlying LCS is composed from these LCS's by means of the (reversible) LCS-to-syntax mappings

[2]For discussion of projection to maximal level by the syntactic component of the system, see [Dorr, 1987]. In a nutshell, **X-MAX** refers to the **XP** phrase that contains a word of category **X**.

[3]In this case, there is only one possible syntactic tree; however, if the structure were ambiguous, other possibilities would be returned. The *e* elements under C and I are syntactic positions for which there is no overt lexical material.

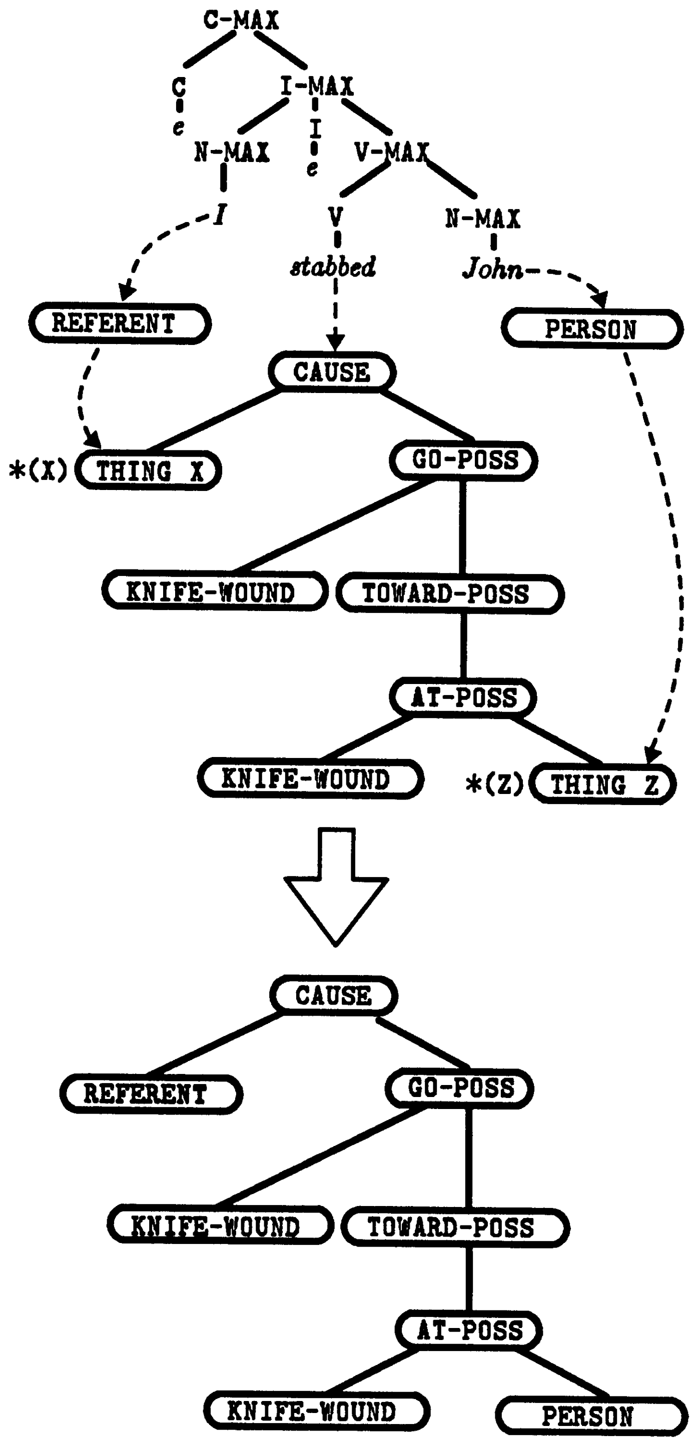


Figure 13: Mapping Syntactic Tree Positions to LCS Argument Positions; Derivation of the Composed LCS for the *stab* Event

1.	Lexically select target language word.
2.	Syntactically realize target language word.
3.	Realize arguments of target language word, if any.

Figure 14: Procedure for Lexical Selection and Syntactic Realization

described in the last section. For example, the LCS **REFERENT** is selected for the word *I* and the LCS **PERSON** is selected for the word *John*; during LCS composition, the **REFERENT** is mapped by the generalized linking routine to the external position **X** of *stab*, and the **PERSON** is mapped by the same routine to the internal position **Z** of *stab*. Figure 13 shows the mapping from the syntactic tree to the LCS argument positions in the definition of *stab*; the result of this mapping is the composed LCS as shown.

Once this LCS has been composed, the third module of the LCS component performs *lexical selection* and *syntactic realization* to produce the final target-language tree and sentence. These two operations are the first and second steps executed by this module as shown in figure 14. Note that the third step, argument realization, is actually a recursive call to this procedure: arguments of a target language word are realized in the same way that the target language word was realized. We will now examine the lexical selection and syntactic realization steps in more detail, and then go on to show how these steps are applied to the current example.

5 Lexical Selection and Syntactic Realization

Lexical selection is the task of choosing target-language words that accurately reflect the meaning of the corresponding source-language words. Syntactic realization is the task of instantiating an appropriate syntactic structure for the LCS composed from the source-language words. The main difficulty associated with both of these tasks is that lexical words and their arguments may not be syntactically realized in the same way across languages. Some of the divergences that arise during translation are shown in figure 15. Despite these divergences, LCS descriptions provide the abstraction necessary for lexical selection and syntactic realization because they allow lexical entries to specify syntactic information separately from conceptual information. The rest of this report will discuss the *stab* example; other examples are presented in [Dorr, 1989].

The lexical-selection procedure maps an LCS node to an appropriate lexi-

Divergence Type	*Translation Example*
Categorial	I am hungry ⇕ Tengo hambre (I have hunger)
Thematic	I like the book ⇕ Me gusta el libro (The book pleases me)
Conflational	John broke into the room ⇕ Juan forzó la entrada al cuarto (John forced entry to the room)
Structural	I saw John ⇕ Vi a Juan (I saw to John)
Lexical	I stabbed John ⇕ Yo di cuchilladas a Juan (I gave knife-wounds to John)

Figure 15: Traditionally Difficult Problems for Machine Translation

cal root word from a target-language possibility set, and then recursively does the same for each of the arguments of that node. The syntactic-realization procedure projects each target-language root word to its phrasal level and attaches it according to the positioning requirements of the lexical root word that selects it.

To clarify this process, we return to our translation example (3). During the translation of this sentence, we run into several of the above-mentioned syntactic divergences. In particular, the words *stab* and *dar* are lexically divergent, the unrealized (conflated) argument *knife-wound* is realized as *cuchilladas*, and the thematic recipient *John* has the structurally divergent realization *a Juan*. Figure 16 shows the contrasting syntactic trees for the source- and target-language sentences. We will now examine how the LCS representation aids the lexical selection and syntactic realization processes for this example.

Refer to figure 10 for the LCS definitions associated with the English and

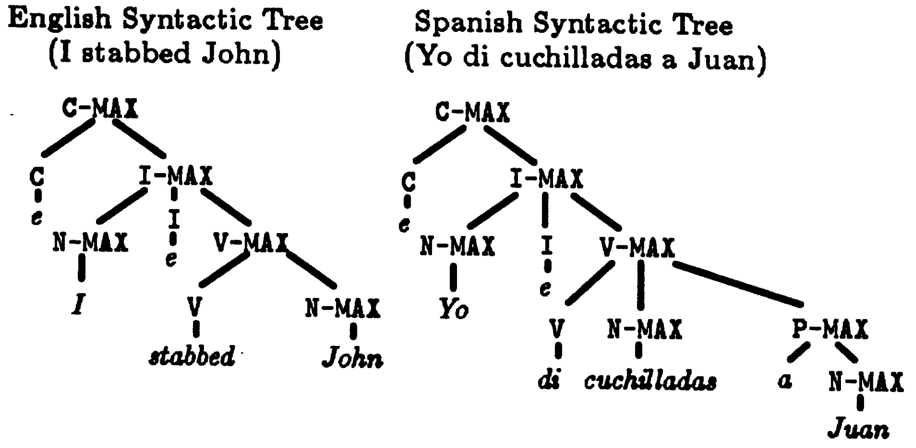


Figure 16: English and Spanish Syntactic Trees for the *stab* Event

Spanish tokens in this sentence. Once the LCS for this sentence has been composed (see figure 13), the lexical selection procedure (step 1 of figure 14) must choose the appropriate Spanish root word by matching the underlying LCS not only at top level, but also at all lower levels. In general, there are two types of LCS nodes that are taken into consideration during the matching process of lexical selection. The more general nodes (*e.g.*, BE-IDENT, GO-POSS, *etc.*) allow the matcher to determine the LCS class of the target-language term; the more specific nodes (*e.g.*, FORCEFULLY, LIKINGLY, *etc.*) are used for final convergence on a particular target-language term such as *like* as opposed to *love*, and *force* as opposed to *cause*. Figure 17 shows how the composed LCS is disassembled into the component LCS definitions associated with target-language words; the argument positions of these LCS definitions map to positions in the syntactic tree that is generated (as shown) for this example.

The list of Spanish root words corresponding to the top-level CAUSE LCS is shown in figure 18 (a). Of these root word possibilities, a smaller set matches the LCS GO-POSS at the next level as shown in figure 18 (b). Note, for example, that the root word *forzar* (the literal translation of *force*) is eliminated as a possibility at this level because its associated LCS does not contain GO-POSS. Finally, at the next level (*i.e.*, the TOWARD-POSS level), only the root word *dar* matches, as shown in figure 18 (c). Thus, this root is selected to be the word that will be projected. Note that lexical selection of this root word involves only three iterations of the matcher in a search space that is reasonably small. Although the compositional nature of the LCS's allows for a large set of words to be defined, the number of primitives to search through at each level of the matching process is bounded by the number of primitives in the system (approximately 25).

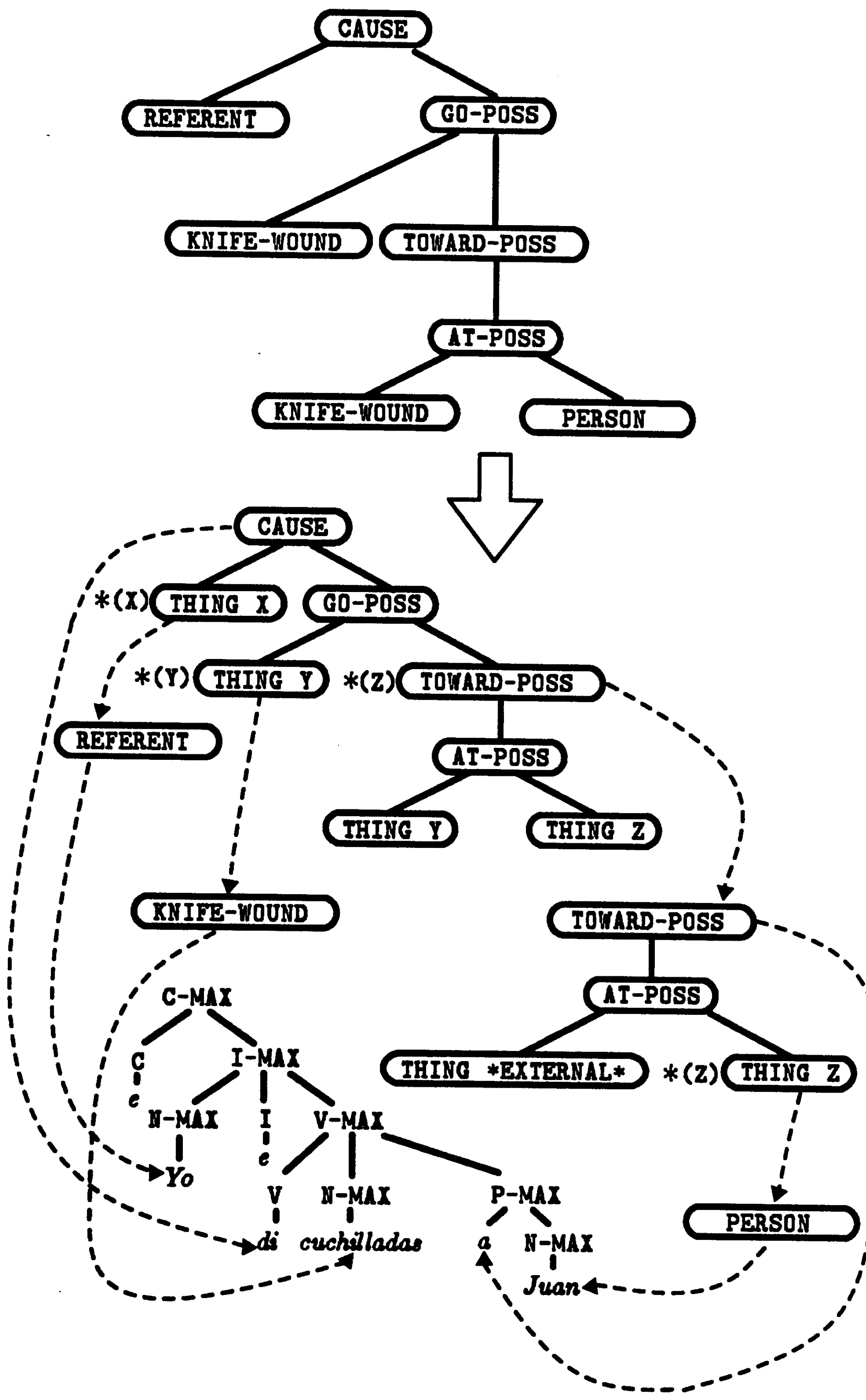


Figure 17: Disassembly of the Composed LCS for the *stab* Event; Mapping LCS Argument Positions to Syntactic Tree Positions

(a) Spanish root words corresponding to the top-level LCS CAUSE: <ROOT-WORD comprar V>, <ROOT-WORD contar V>, <ROOT-WORD contribuir V>, <ROOT-WORD cortar V>, <ROOT-WORD dar V>, <ROOT-WORD decir V>, <ROOT-WORD embadurnar V>, <ROOT-WORD escribir V>, <ROOT-WORD forzar V>, <ROOT-WORD matar V>, <ROOT-WORD pintarrajear V>, <ROOT-WORD recibir V>, <ROOT-WORD romper V>, <ROOT-WORD vender V>
(b) Spanish root words corresponding to the top-level LCS CAUSE that includes GO-POSS: <ROOT-WORD comprar V>, <ROOT-WORD contribuir V>, <ROOT-WORD cortar V>, <ROOT-WORD dar V>, <ROOT-WORD escribir V>, <ROOT-WORD recibir V>, <ROOT-WORD vender V>
(c) Spanish root word corresponding to the LCS GO-POSS that includes TOWARD-POSS: <ROOT-WORD dar V>

Figure 18: Spanish Root Word Possibilities for the *stab* Event

Once the appropriate root word is chosen, the CSR function is called, and CAUSE (which is an EVENT) is realized as a verb phrase (V-MAX). Next, the system must lexically select the arguments of the root word *dar*. The '*' marker in the LCS-to-syntax mapping is used to determine the level at which each argument will be realized. The first argument that gets realized is the subject of *dar* which is the REFERENT node. The first-person singular pronominal corresponding to this node is *yo* in Spanish; thus, this root word is selected to be realized as the subject. The next step is to realize the objects of *dar*. Note that unlike the *stab* definition, the *dar* definition requires the Y argument to be overtly realized (see figure 10); thus, the system performs an "inverse conflation" in order to arrive at the target-language realization for this example. Because the composed LCS contains the argument KNIFE-WOUND, this becomes the token that will be overtly realized for the Y variable in the Spanish LCS. Now the lexical selection routine is called recursively to determine the possible target-language words for the KNIFE-WOUND argument. As shown in figure 10, the word *cuchillada* is defined to be a KNIFE-WOUND, so this is the root word chosen to be the direct object of the main verb. The CSR function is then called to realize KNIFE-WOUND (which is a THING) as a noun phrase (N-MAX).

Next, the second object of *dar* must be realized. Note that this argument must be realized at the level of TOWARD-POSS (denoted by '*' in figure 10).

Since the word *a* is defined as TOWARD-POSS, this word is chosen as the target lexical item. Then the CSR function is called on TOWARD-POSS (which is a PATH) and the argument is realized as a prepositional phrase (P-MAX).

Finally, the lexical selection procedure is called once again to realize the Z variable that corresponds to the PERSON argument. This argument is realized as the N-MAX *Juan*, and the Spanish syntactic tree of figure 17 is generated. The final output sentence is produced by reading off the leaves of the syntactic tree.

6 Limitations and Future Work

UNITRAN was deliberately designed to operate on one sentence at a time, and as such, there are a number of inherent limitations. For example, the system does not incorporate context or domain knowledge; thus, it cannot use discourse, situational expectations, or domain information in order to generate a sentence. Consequently, there are a number of capabilities found in other systems that cannot be reproduced here including external pronominal reference (*e.g.*, MUMBLE, [McDonald, 1983] and [McDonald, 1987]), paraphrasing (*e.g.*, MOPTRANS, [Lytinen & Schank, 1982]), story telling (*e.g.*, SAM, [Schank & Abelson, 1977] and [Cullingford, 1986]), interactive question-answering (*e.g.*, TEXT, [Mckeown, 1985]), *etc.* This is not to say that issues of context and domain knowledge should be ignored; on the contrary, these types of knowledge may be the next step in the evolution of the UNITRAN system.

An additional limitation of the LCS approach is the potential for generating several target-language possibilities for a given lexical-semantic primitive. It is possible that the LCS-matching procedure will not adequately cut down the target-language possibilities during the mapping from LCS to lexical items. For example, there are many open-ended classes of words (in particular, proper and common nouns, and certain adjectives and adverbs) that are not distinguishable by their LCS's. This is because LCS's provide an underlying representation of predicates and their arguments; any lexical item that does not exhibit a predicate-argument relationship must be translated by other means. Thus, if the possibility list is still quite large (more than two or three lexical items) after LCS-matching routines have finished the lexical selection process, a direct-mapping routine is used instead for lexicalization. That is, certain lexical-items (*John*, *book*, *red*, *quickly*, etc.) may be selected on the basis of a direct mapping to the surface form. Pustejovsky & Nirenburg [1987] provide an elegant approach to generation of open-class lexical items based on focus information. Because the system described here does not include a model of discourse, the direct-mapping technique is used for such problematic cases.

Another limitation of the system as it stands is that the notion of *aspect* is not represented in the LCS structures. For example, there is no way to establish whether an event is prolonged, repeated, instantaneous, *etc.* Thus, in the *stab* example, there is no way to determine how many times the stabbing action occurred. As it turns out, the Spanish surface realization relies on this missing information. The translation of the repetitive version of *stab* is the surface form *dar cuchilladas* (the plural form of knife-wound), whereas the translation of the non-repetitive version is the surface form *dar una cuchillada* (the singular form of knife-wound). Jackendoff *does* try to include the notion of aspect in some cases. For example, the lexical-semantic token BE-CIRC allows the progressive aspect to be expressed. However, there is no way to determine the appropriate aspect in the general case, so the system arbitrarily chooses a target-language word when such an ambiguity arises. Superimposing a system of aspect such as that of Tenny [1989] could prove to be useful in the future.

7 Summary

The UNITRAN system is implemented in Common Lisp and is currently running on a Symbolics 3600 series machine. The LCS approach has been shown to be valuable for machine translation because it facilitates two crucial translation operations: lexical selection and syntactic realization. Furthermore, because it is compositional in nature, the LCS representation aids in tackling some of the more difficult problems in machine translation. In addition, the lexical word classes derived from the LCS hierarchy reduce proliferation of lexical redundancy by providing a means of relating word senses without restating entire definitions.

We have seen that the definition of a potentially large (theoretically infinite) set of words is supported by the ability to combine the same lexical-semantic primitives in an indefinite number of ways. However, as shown in the last example, the search space for root word selection is not explosive because there are only a small number of primitives that must be searched at each level of the matching procedure.

LCS descriptions seem to provide the abstraction necessary for selecting appropriate target-language terms with minimal dependence on syntax, and they also provide the necessary mechanism for realizing arguments without regard to conceptual considerations. In particular, lexical entries are divided into two levels of description, lexical-semantic and syntactic; the former is used for lexically selecting arguments, and the latter is used for syntactically "fitting" these arguments into a predicate-argument structure.

The approach presented here tries to incorporate some of the more promising syntactic and semantic aspects of existing translation systems. Specifically, the model incorporates structural information for realization and positioning of arguments. Unlike direct-replacement and entirely syntactic-based approaches, however, it avoids non-compositional direct-mapping word selection. In addition, the model has the ability to select target terms on the basis of compositional properties. Unlike many semantic-based approaches, however, it does not appeal to context-dependent routines, and it does not entirely abandon syntactic considerations for selection and realization of root words and their associated arguments.

In summary, this report has shown that the UNITRAN system handles a number of traditionally difficult cases for machine translation by providing a general compositional representation that allows for syntactic variation across languages.

8 References

Chomsky, Noam A. (1986) *Knowledge of Language: Its Nature, Origin and Use*, MIT Press, Cambridge, MA.

Cullingford, Richard E. (1986) *Natural Language Processing: A Knowledge-Engineering Approach*, Rowman and Littlefield, Totowa, New Jersey.

Dorr, Bonnie J. (1987) "UNITRAN: A Principle-Based Approach to Machine Translation," AI Technical Report 1000, Master of Science thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Dorr, Bonnie J. (1989) "Lexical Conceptual Structure and Generation in Machine Translation," *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, Michigan.

Dorr, Bonnie J. (forthcoming) "Lexical Conceptual Structure and Machine Translation," Ph.D. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Hale, Kenneth and Jay Keyser (1986) "Some Transitivity Alternations in English," Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA, Lexicon Project Working Papers #7.

Hale, Kenneth and M. Laughren (1983) "Warlpiri Lexicon Project: Warlpiri Dictionary Entries," Massachusetts Institute of Technology, Cambridge, MA, Warlpiri Lexicon Project.

Jackendoff, Ray S. (1983) *Semantics and Cognition*, MIT Press, Cambridge, MA.

Lytinen, Steven and Roger Schank (1982) "Representation and Translation," Department of Computer Science, Yale University, New Haven, CT, Technical Report 234.

McDonald, David D. (1983) "Natural Language Generation as a Computational Problem," in *Computational Models of Discourse*, Brady, Michael and Robert C. Berwick (eds.), MIT Press, Cambridge, MA, 209–265.

McDonald, David D. (1987) "Natural Language Generation: Complexities and Techniques," in *Machine Translation: Theoretical and Methodological Issues*, Sergei Nirenburg (ed.), Cambridge University Press, Cambridge, England.

McKeown, Kathleen (1985) *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*, Cambridge University Press, Cambridge, England.

Pustejovsky, James and Sergei Nirenburg (1987) "Lexical Selection in the Process of Language Generation," *Proceedings of the 25th Annual Conference of the Association for Computational Linguistics*, Stanford University, Stanford, CA, 201–206.

Rappaport, Malka and Beth Levin (1986) "What to Do with Theta-Roles," Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA, Lexicon Project Working Papers #11.

Schank, Roger C. and Robert Abelson (1977) *Scripts, Plans, Goals, and Understanding*, Lawrence Erlbaum Associates, Inc., Hillsdale, NJ.