On Beyond Syntax: Use of Lexical Conceptual Structure for Intelligent Tutoring

Bonnie Dorr, James Hendler,
Scott Blanksteen, and Barrie Migdaloff
Department of Computer Science
University of Maryland
A.V. Williams Building
College Park, Maryland 20742

Abstract: We describe the use of an LCS-based semantics for question-answering exercises in foreign language training. We start by reviewing the LCS model and show how this representation can be used to support a question answering lesson as well as a limited domain discourse. An authoring tool for entering the relevant semantic knowledge is also described.

1 Introduction

The chapters in this book result from a workshop which presented a number of state-of-the-art projects in Intelligent Computer Aided Language Learning (ICALL). At that workshop, several different speakers stated that the study of syntactic processing was at a mature enough state that it could be reliably used as a basis for tutoring systems. Semantic knowledge, however, was deemed to be less well understood and not reliable for use outside of extremely limited domains such as graphics microworlds or "adventure game" level discourse. Further, authoring of semantic knowledge was dismissed as not currently available.

It is our contention that the above is simply wrong. Current advances in lexical semantics make it possible to provide a wide range of new capabilities. In particular, recent work in computational linguistics (Dorr (1992, 1993a,b), Jackendoff (1983, 1990)) has shown that a lexically based semantics, tied relatively closely to grammatical structures, can be used to create an interlingua that can support activities such as machine translation and language tutoring. While it is the case that these lexical structures fall short of providing full semantic inferencing, and while knowledge based systems are still limited in the domains for which such inferencing can be supported, these lexical semantics provide a robust basis for the development of tutoring functions and an analysis that is more useful than syntactic analysis.

A simple example can easily show the need for going beyond syntactic processing in computer-aided language tutoring situations. Given a syntactically based language tutor, such as the Bridge tutor developed by a team of researchers from SAIC, University of Maryland and the Army Research Institute (Sams (1993)), student inputs can be analyzed for syntactic correctness, but not for whether the answer

is appropriate to the question. In one lesson, Bridge presents the student with a map, and asks the student to provide a sentence in the target language (German in this example) which answers the question:

TUTOR> Wie fährt man von Heidelberg nach Frankfurt?
(How does one drive from Heidelberg to Frankfurt?)

The student's answer is processed through a syntactic parser. If the student gives an appropriate answer, for example "Folgen Sie der Autobahn im Norden (*You take the Autobahn to the north*)," the system attests that the answer is syntactically correct, and the student continues on. If there is a syntactic error, the student is corrected in various ways, and the lesson continues.

However, consider what happens if the student types "Ich bin Berliner (*I am from Berlin*)" in answer to the question above. As there are no syntactic mistakes in the sentence, the student is rewarded for a correct answer! Clearly, the system has missed the fact that the answer was inappropriate for the question asked. Catching such problems is quite difficult, however, since the student can produce many sentences that correctly answer the question, such as: "You take the Autobahn northwards," "You go north on the Autobahn," "You take a Autobahn to the north," etc. Thus, the system must have some model of the semantics of the sentence if it is to correctly recognize the cases where the student's answer has the same meaning as the instructor's expected response.

In the sections that follow, we will discuss the semantic models used by a tutoring project which is being developed jointly by the University of Maryland and Micro Analysis and Design Corp under sponsorship from the Army Research Institute. The general goal of this tutor is to allow a wide range of lessons to be developed for use in language training.[1] In addition to a number of authoring tools, speech and written display lessons, and various types of language drills, the tutor will support an authorable question answering system, and a limited-domain dialog lesson. To support these goals, we are developing a natural language processing system which can encode the wide range of lexical and semantic knowledge that is relevant for understanding utterances, whether they are placed in an overt lesson or dialog context.

In this chapter we discuss the semantic tools that let us go beyond the syntactic capabilities of tutors such as Bridge. We start by reviewing the particular lexical semantic model which we use (section 2). We show how this representation supports a question answering system (section 3) and a limited domain discourse (section 5). An authoring tool for entering the relevant semantic knowledge is also described (section 4). We conclude by discussing future work, particularly focusing on developing a better integration between the lexical semantics and deeper knowledge-based inferencing, which will be needed to improve these capabilities in the future (section 6).

[1] The first languages targeted for this system are Arabic and Spanish.

2 Lexical Conceptual Structure

One of the types of knowledge that must be captured in the tutor is linguistic knowledge at the level of the lexicon, which covers a wide range of information types such as verbal subcategorization for events (e.g., that a transitive verb such as "hit" occurs with an object noun phrase), featural information (e.g., that the direct object of a verb such as "frighten" is animate), thematic information (e.g., that "John" is the agent in "John hit the ball"), and lexical-semantic information (e.g., spatial verbs such as "throw" are conceptually distinct from verbs of possession such as "give"). By modularizing the lexicon, we treat each information type separately, thus allowing us to vary the degree of dependence on each level so that we can address the question of how much knowledge is necessary for the success of the tutoring system.

The most intricate component of lexical knowledge is the lexical-semantic information, which we encode in the form of Lexical Conceptual Structure (LCS) as formulated by Dorr (1992, 1993a,b) based on work by Jackendoff (1983, 1990). This representation abstracts away from syntax just far enough to enable language independent encoding, while retaining enough structure to be sensitive to the requirements for multi-lingual processing. The central idea of LCS is that the human language can be modeled using a uniform internal representation of conceptual information with a few parameters that can be toggled one way or another to obtain various languages. The versatility of the LCS allows us to reuse labor-intensive features of the lexicon across languages. We perform natural language analysis by mapping the syntactic representation of the input sentences into this internal representation; the result is used later as the basis for semantic interpretation. This technique produces more accurate analyses than naive syntactic-tree matching approaches, which tend to be much more limited in coverage.

A central issue that arises with respect to the use of a language-independent representation in any multilingual system is that of defining a set of primitives to represent cross-linguistic phenomena. Because it is generally difficult to define such a set, many researchers have abandoned the use of an intermediate representation in multilingual applications. (See, for example, Vauquois and Boitet (1985).) However, recently, there has been a resurgence of interest in the area of lexical representation and organization (with special reference to verbs) that has initiated an ongoing effort to delimit the classes of lexical knowledge required to process natural language. (See, e.g., Grimshaw (1990), Hale and Keyser (1986a, 1986b, 1989), Hale and Laughren (1983), Jackendoff (1983, 1990), Levin and Rappaport (1986), Levin (1985, 1993), Pustejovsky (1988, 1989, 1991), Rappaport et al. (1987), Rappaport and Levin (1988), Olsen (1991), and Zubizarreta (1982, 1987).) As a result of this effort, it has become increasingly more feasible to isolate the components of meaning common to verbs participating in particular classes. These components of meaning can then be used to determine the lexical representation of verbs across languages.

The LCS approach views semantic representation as a subset of conceptual structure, i.e., the language of mental representation. Jackendoff's approach includes *types* such as Event and State, which are specialized into *primitives* such as GO, STAY, BE, GO-EXT, and ORIENT. As an example of how the primitive GO is

used to represent sentence semantics, consider the following sentence:

(1) (i) The ball rolled toward Beth.

(ii) [$_{\text{Event}}$ GO ([$_{\text{Thing}}$ BALL],
[$_{\text{Path}}$ TOWARD ([$_{\text{Position}}$ AT ([$_{\text{Thing}}$ BALL], [$_{\text{Thing}}$ BETH])])])]

This representation illustrates one dimension (i.e., the *spatial* dimension) of Jackendoff's representation. Another dimension is the *causal* dimension, which includes the primitives CAUSE and LET. These primitives take a Thing and an Event as arguments. Thus, we could embed the structure shown in (1)(ii) within a causative construction:

(2) (i) John rolled the ball toward Beth.

(ii) [$_{\text{Event}}$ CAUSE
([$_{\text{Thing}}$ JOHN],
[$_{\text{Event}}$ GO ([$_{\text{Thing}}$ BALL],
[$_{\text{Path}}$ TOWARD [$_{\text{Position}}$ AT ([$_{\text{Thing}}$ BALL], [$_{\text{Thing}}$ BETH])])])])

Jackendoff includes a third dimension by introducing the notion of *field*. This dimension extends the semantic coverage of spatially oriented primitives to other domains such as Possessional, Temporal, Identificational, Circumstantial, and Existential.[2] For example, the primitive GO_{Poss} refers to a GO event in the Possessional field as in the following sentence:

(3) (i) Beth received the doll.

(ii) [$_{\text{Event}}$ GO_{Poss}
([$_{\text{Thing}}$ DOLL],
[$_{\text{Path}}$ TO_{Poss} ([$_{\text{Position}}$ AT_{Poss} ([$_{\text{Thing}}$ DOLL], [$_{\text{Thing}}$ BETH])])])]

To further illustrate the notion of field, the GO primitive can be used in the Temporal and Identificational fields:

(4) (i) The meeting went from 2:00 to 4:00.

(ii) [$_{\text{Event}}$ GO_{Temp}
([$_{\text{Thing}}$ MEETING],
[$_{\text{Path}}$ $\text{FROM}_{\text{Temp}}$ ([$_{\text{Position}}$ AT_{Temp} ([$_{\text{Thing}}$ MEETING], [$_{\text{Time}}$ 2:00])])]
[$_{\text{Path}}$ TO_{Temp} ([$_{\text{Position}}$ AT_{Temp} ([$_{\text{Thing}}$ MEETING], [$_{\text{Time}}$ 4:00])])])]

(5) (i) The frog turned into a prince.

(ii) [$_{\text{Event}}$ GO_{Ident}
([$_{\text{Thing}}$ FROG],
[$_{\text{Path}}$ TO_{Ident} ([$_{\text{Position}}$ AT_{Ident} ([$_{\text{Thing}}$ FROG], [$_{\text{Thing}}$ PRINCE])])])]

Figure 1 shows a subset of the types and primitives that are currently used in the LCS scheme. Figure 2 shows the different types of sentences that can be represented by this system of primitives and fields. It should be noted that, although the LCS

[2] As we will see shortly, the label Loc has been adopted to distinguish the spatial field from the non-spatial fields. Note that the spatial field is used to denote the primitives that fall in the spatial dimension. Jackendoff argues that spatial primitives are more fundamental than those of other domains (e.g., Possessional). Thus, spatial primitives have their own special status as an independent dimension.

Type	**Primitives**
Event	CAUSE, LET, GO, STAY
State	BE, GO-EXT, ORIENT
Position	AT, IN, ON
Path	TO, FROM, TOWARD, AWAY-FROM, VIA
Thing	BOOK, PERSON, REFERENT, KNIFE-WOUND, KNIFE, SHARP-OBJECT, WOUND, FOOT, CURRENCY, PAINT, FLUID, ROOM, SURFACE, WALL, HOUSE, BALL, DOLL, MEETING, FROG
Property	TIRED, HUNGRY, PLEASED, BROKEN, ASLEEP, DEAD, STRETCHED, HAPPY, RED, HOT, FAR, BIG, EASY, CERTAIN
Location	HERE, THERE, LEFT, RIGHT, UP, DOWN
Time	TODAY, SATURDAY, 2:00, 4:00
Manner	FORCEFULLY, LIKINGLY, WELL, QUICKLY, DANCINGLY, SEEMINGLY, HAPPILY, LOVINGLY, PLEASINGLY, GIFTINGLY, UPWARD, DOWNWARD, WITHIN, HABITUALLY

Figure 1: LCS primitives and Types

Primitive	*Primitive-Field*	*Example*
GO	GO_{Poss}	Beth received the doll.
	GO_{Ident}	Elise became a mother.
	GO_{Temp}	The meeting went from 2:00 to 4:00.
	GO_{Loc}	We moved the statue from the park to the zoo.
	GO_{Circ}	John started shipping goods to California.
	GO_{Exist}	John built a house.
STAY	$STAY_{Poss}$	Amy kept the doll.
	$STAY_{Ident}$	The coach remained a jerk.
	$STAY_{Temp}$	We kept the meeting at noon.
	$STAY_{Loc}$	We kept the statue in the park.
	$STAY_{Circ}$	John kept shipping goods to California.
	$STAY_{Exist}$	The situation persisted.
BE	BE_{Poss}	The doll belongs to Beth.
	BE_{Ident}	Elise is a pianist.
	BE_{Temp}	The meeting is at noon.
	BE_{Loc}	The statue is in the park.
	BE_{Circ}	John is shipping goods to California.
	BE_{Exist}	Descartes exists.
GO-EXT	$GO\text{-}EXT_{Ident}$	Our clients range from psychiatrists to psychopaths.
	$GO\text{-}EXT_{Temp}$	The meeting lasted from noon to night.
	$GO\text{-}EXT_{Loc}$	The road went from Boston to Albany.
ORIENT	$ORIENT_{Loc}$	The sign points to Philadelphia.
	$ORIENT_{Circ}$	John intended to ship goods to California.

Figure 2: Sentences Represented by Event and State primitives of the LCS

Class of Verb	Examples
position	be, remain, ...
change of position	fall, throw, drop, change, move, slide, float, roll, fly, bounce, move, drop, turn, rotate, shift, ...
directed motion	enter, break into, bring, carry, remove, come, go, leave, arrive, descend, ascend, put, raise, lower, ...
motion with manner	sail, walk, stroll, jog, march, gallop, jump, float, dance, run, skip, ...
exchange	buy, sell, trade, ...
physical state	be, remain, keep, leave, ...
change of physical state	open, close, melt, redden, soften, break, crack, freeze, harden, dry, whiten, grow, change, become, ...
orientation	point, aim, face, ...
existence	exist, build, grow, shape, make, whittle, spin, carve, weave, bake, fashion, create, appear, disappear, reappear, persist, ...
circumstance	be, start, stop, continue, keep, exempt, ...
range	go, last, extend, intend, aim, range, ...
change of ownership	give, take, receive, relinquish, borrow, lend, loan, steal, ...
ownership	belong, remain, keep, own, ...
ingestion	eat, drink, smoke, gobble, munch, sip, ...
psychological state	like, fear, admire, detest, despise, enjoy, esteem, hate, honor, love please, scare, amuse, astonish, bore, surprise, stun, terrify, thrill, ...
perception and communication	see, hear, smell, feel, look, watch, listen, learn, sniff, show, tell, talk, speak, shout, whisper, scream, ...
mental process	know, learn, ...
cost	cost, charge, ...
load/spray	smear, load, cram, spray, stuff, pile, stack, splash, ...
contact/effect	cut, stab, crush, smash, pierce, bite, shoot, spear, ...

Figure 3: Linguistic Classes from Levin (1985, 1993) implemented in LCS Framework

representation appears to be somewhat "English-like," it is only superficially so by virtue of the labels of the primitives (e.g., GO, TO, etc.) that are used in the representation. These primitives were chosen on the basis of an extensive cross-linguistic investigation, though they may be labeled and used in a fashion that appears to be modeled after a particular language. The verb classes (based on work by Levin (1985, 1993)) that are currently accommodated are shown in figure 3. It is expected that this verb classification scheme will need further refinement as more properties of verbs are identified.

To illustrate the use of this representation in the lexicon, consider the following example:

(6) E: I like Mary
S: María me gusta
(Mary (to) me pleases)

The language-independent representation for this example looks like the following:

(7) $[_{\text{State}}\ \text{BE}_{\text{Ident}}\ ([_{\text{Thing}}\ \text{I}],$
$[_{\text{Position}}\ \text{AT}_{\text{Ident}}\ ([_{\text{Thing}}\ \text{I}],\ [_{\text{Thing}}\ \text{MARY}])],$
$[_{\text{Manner}}\ \text{LIKINGLY}])]$

This representation roughly means "I am in an identificational state LIKINGLY with respect to Mary." Both the Spanish and English sentences are based on this

representation; the syntactic distinction (i.e., the subject-object reversal) is captured by means of parameterization in the lexicon:

(8) (i) **Lexical Entry for like**:
$[_{\text{State}}\ \text{BE}_{\text{Ident}}\ ([_{\text{Thing}}\ \text{:EXT W}],$
$[_{\text{Position}}\ \text{AT}_{\text{Ident}}\ ([_{\text{Thing}}\ \text{W}], [_{\text{Thing}}\ \text{:INT Z}])],$
$[_{\text{Manner}}\ \text{LIKINGLY}])]$

(ii) **Lexical Entry for gustar**:
$[_{\text{State}}\ \text{BE}_{\text{Ident}}\ ([_{\text{Thing}}\ \text{:INT W}],$
$[_{\text{Position}}\ \text{AT}_{\text{Ident}}\ ([_{\text{Thing}}\ \text{W}], [_{\text{Thing}}\ \text{:EXT Z}])],$
$[_{\text{Manner}}\ \text{LIKINGLY}])]$

The :INT/:EXT markers are examples of lexical parameterization that allow the system to account for the subject-object reversal of the *like-gustar* example.

The LCS representation has been developed on the basis of a study of cross-linguistic distinctions such as that of (6). This decompositional approach to describing linguistic constructions is consistent with the philosophy behind well-grounded lexical-semantic theories such as *meaning-text theory* (MTT) by Mel'čuk and Polguère (1987, p. 266); that is, rather than postulating a set of semantic primitives *a priori*, the intention is to discover them by means of a "painstaking process of semantic decomposition applied to thousands of actual lexical items." Given that recent research has made it increasingly more feasible to isolate components of verb meaning, this approach to the construction of a language-independent representation is no longer impractical. Note that this is in direct contrast to other approaches that employ primitives that are chosen *a priori* such as the conceptual dependency approach by Schank (1972, 1973, 1975) and Schank and Abelson (1977), for which there has never been a cross-linguistic investigation into the applicability of the primitives.

A final point to be made about the current representation is that it is intended to include those aspects of lexical knowledge related to argument structure, not "deeper" notions of meaning such as aspectual, contextual, domain, and world knowledge. While these "deeper" notions are indisputably necessary for a general solution to language understanding, previous approaches that have employed deep semantic representations say very little about how to map the representation systematically to the surface syntactic structure, especially in the context of cross-linguistic distinctions such as that of (6). These approaches pay a high price for incorporating too much "deeper" knowledge without preserving structurally defined lexical-semantic information. This is not to say that research should head in the direction of "shallow" representations (e.g., see Sharp (1985)). The current approach attempts to achieve a middle ground between representations that encode too much information and those that encode too little. Specifically, the current representation captures parametric information that accommodates cross-linguistic distinctions without losing the systematic relation between the language-independent representation and the syntactic structure. Moreover, the representation lends itself readily to the specification of processing components that operate uniformly across different languages.

3 Using the LCS for Question Answering Lessons

As discussed in the introduction, the LCS is a critical component of our approach to matching student answers in the context of a question answering lesson provided by the foreign language tutor. This lesson requires the student to answer questions about a short foreign language passage. In order to inform the student whether a question has been answered correctly, the author of the lesson must provide the desired response in advance. The system parses and semantically analyzes this response into a corresponding LCS representation which is then prestored in a database of possible responses. Once the question answering lesson is activated, each of the student's responses is parsed and semantically analyzed into a LCS representation which is checked for a match against the corresponding prestored LCS representation. The student is then informed as to whether the question has been answered correctly depending on how closely the student's response LCS matches the author's prestored LCS.

The overall processing design used to support question answering exercises is shown in figure 4. The analysis component of the system includes morphological, syntactic, and semantic processing that transforms the student's input sentence (i.e., the response to the tutor's question) into a language-independent representation. The syntactic analyzer makes use of Government Binding principles (developed by Chomsky (1981, 1986) and his followers) that are parameterized for the particular language in question. Thus, the analyzer is able to function in both Arabic and Spanish, given the appropriate lexicon. The parse tree produced by the syntactic analyzer is passed to the semantic analysis/composition component, and the language-independent LCS representation is generated. This representation is then passed to the LCS Matcher which checks the answer for a match against the desired answer that was prestored by the author. The degree to which the match succeeds is then determined, and this is used to generate the tutor output (i.e., confirmation of (in)correctness).

3.1 LCS Matching

To support question answering lessons, a student's input must be matched against the instructor's stored answer to see if the answer is appropriate. In this section we briefly describe the matcher and its output; in the next, we discuss how it is used in the question answering context.

The matcher is a relatively straightforward algorithm, which performs a recursive descent through two LCSs determining where they are the same and where they are different. The matcher starts by determining whether two LCS structures contain the same primitive. If they do, it determines for each of the fields whether they are omitted or, if present, whether they contain the same fillers. If all of the fields in the stored answer are correctly matched with the student's answer, then the match is said to be correct. If extra information is provided by the student, this is identified as well. The return from the matcher identifies whether the match is correct, and what fields are missing or extra.

As an example, consider what happens in a lesson if the teacher has specified that a correct answer is "John ran to the house." This is processed by the system

Figure 4: Diagram of Processing Flow for Question/Answering

to produce the following LCS:

(9) [$_{\text{Event}}$ GO_{Loc}
([$_{\text{Thing}}$ JOHN],
[$_{\text{Path}}$ TO_{Loc}
([$_{\text{Position}}$ AT_{Loc} ([$_{\text{Thing}}$ JOHN], [$_{\text{Property}}$ HOUSE])])],
[$_{\text{Manner}}$ RUNNINGLY])]

This is stored by the tutor and then later matched against the student's answer. If the student types "John went to the house" the system must determine if these two match. The student's sentence is processed and the following LCS structure is produced:

(10) [$_{\text{Event}}$ GO_{Loc}
([$_{\text{Thing}}$ JOHN],
[$_{\text{Path}}$ TO_{Loc}
([$_{\text{Position}}$ AT_{Loc} ([$_{\text{Thing}}$ JOHN], [$_{\text{Property}}$ HOUSE])])])]

The matcher compares these two, and produces a data structure representing the output:

```
Missing :
 MANNER RUNNINGLY

Extra :
 NIL

INCORRECT answer.
```

This identifies the student's response as an incorrect answer, since the student has omitted the fact that John was running.

If we reverse the situation, with the teacher storing as the correct answer "John went to the house" and the student typing "John ran to the house," the situation would be different. In this case, the matcher would return the following:

```
Missing :
 NIL

Extra :
 MANNER RUNNINGLY

CORRECT answer.
```

This time the student has provided information the teacher did not require, but omitted nothing that was required. Thus, the system has identified the answer as correct and recognized that the student provided information about the manner of the going, which was not necessary. The returns from the LCS matcher are passed to the tutor, which turns them into appropriate outputs for the student.

3.2 Question Answering

The question answering lessons are formulated as sets of questions associated with a paragraph (or more) of text in the target language. In an authoring session, the lesson designer enters the passages and the questions, and types a sample appropriate answer for each question.[3] This answer is processed by the system, and the appropriate LCSs are stored. (If the lesson designer wishes to increase the semantic coverage of the system for processing such questions, a LCS editor is provided. This is discussed in section 4.) At lesson time, the student is prompted with these questions, and enters an answer. This answer is processed, and matched as above. If the answer matches, the student continues to the next one. If the answer is identified as an incorrect match, a diagnostic message is prepared by the tutor and shown to the student.

As an example of the use of the LCS processing and matcher to support a lesson, consider the following example of a Spanish passage the student might be expected to read (followed by the corresponding English translation):

(11) (i) Carlos Perez, propietario do Pescadería Perez, fue arrestado ayer en su casa por el asesinato de Juan Rodriguez. El Señor Rodriguez fue encontrado muerto en su apartamento el mes pasado con múltiples heridas de puñaladas en el pecho y estómago. Las huellas tactilares del propietario de la tienda fueron encontrados en el arma asesina, el cual estaba tirado en un cercano basurero.

(ii) Carlos Perez, owner of Perez Fish Market, was arrested yesterday in his home for the murder of Juan Rodriguez. Mr. Rodriguez was found dead in his apartment last month with multiple stab wounds in the chest and stomach. The storeowner's fingerprints were found on the murder weapon, which had been discarded in a nearby trash bin.

The student reads the Spanish passage and is then asked questions about it. An example of a typical question might be the following:

TUTOR> Qué pasó a Juan Rodriguez?
(What happened to Juan Rodriguez?)

Suppose, for the sake of illustration, that the lesson author expects the student's answer to convey the fact that Juan died. Any additional information e.g., that Carlos caused Juan's death or that Juan was stabbed with a knife, would be optionally allowed as well. The author would provide the following sentence:

STUDENT> Juan murió.
(Juan died.)

This would be syntactically parsed and semantically analyzed into the following LCS representation:

(12) $[_{\text{Event}}\ \text{GO}_{\text{Ident}}$
$([_{\text{Thing}}\ \text{JUAN}],$
$[_{\text{Path}}\ \text{TOWARD}_{\text{Ident}}$
$([_{\text{Position}}\ \text{AT}_{\text{Ident}}\ ([_{\text{Thing}}\ \text{JUAN}], [_{\text{Property}}\ \text{DEAD}])])])]$

[3] We currently limit this to only one appropriate answer per question, but expect to relax this limitation in later versions.

This representation would then be stored as the desired response to the current question.

At lesson time, the student would be prompted with the question and expected to type an answer. The student could type any of the following answers:

STUDENT> Juan murió.
(Juan died.)
STUDENT> Carlos le mató a Juan.
(Carlos killed Juan.)
STUDENT> Carlos le asesinó a Juan.
(Carlos murdered Juan.)

The LCS representation corresponding to the first response is the same as the prestored answer provided by the author. The other two are, respectively:

(13) (i) $[_{\text{Event}}$ CAUSE
$([_{\text{Thing}}$ CARLOS],
$[_{\text{Event}}$ GO_{Ident}
$([_{\text{Thing}}$ JUAN],
$[_{\text{Path}}$ $\text{TOWARD}_{\text{Ident}}$
$([_{\text{Position}}$ AT_{Ident} ($[_{\text{Thing}}$ JUAN], $[_{\text{Property}}$ DEAD])])])])]

(ii) $[_{\text{Event}}$ CAUSE
$([_{\text{Thing}}$ CARLOS],
$[_{\text{Event}}$ GO_{Ident}
$([_{\text{Thing}}$ JUAN],
$[_{\text{Path}}$ $\text{TOWARD}_{\text{Ident}}$
$([_{\text{Position}}$ AT_{Ident} ($[_{\text{Thing}}$ JUAN], $[_{\text{Property}}$ DEAD])])])],
$[_{\text{Manner}}$ UNLAWFULLY])]

In all three cases, the matcher would be able to recognize that these are correct answers (each matches the base LCS, although the latter two contain extra information) and the student would proceed.

One advantage of the LCS matching becomes clear in this example. It would be a tedious task if the author were expected to specify, in advance, all possible ways that the student might choose to convey the desired information. Our approach to representing the prestored answer allows the author to type in only the first answer, i.e., *Juan murió*, which is analyzed into a LCS representation; this representation is general enough to match any number of additional answers.

Another advantage concerns the handling of inappropriate answers. If the students types a completely inappropriate answer, such as "Me llamo Juan (*I am called Juan*)" the corresponding LCS would be:

(14) $[_{\text{Event}}$ BE_{Ident} ($[_{\text{Thing}}$ I], $[_{\text{Position}}$ AT_{Ident} ($[_{\text{Thing}}$ I], $[_{\text{Property}}$ JUAN])])]

This is identified by the matcher as being in the wrong primitive class, and rejected as an inappropriate answer. The tutor informs the student that the correct answer is expressed differently and the student tries again.

More subtle errors can also be identified. Consider the case where the student types the following:

STUDENT> Carlos le mató a Juan.
(Carlos killed Juan.)

This answer is almost correct, but the subject and object are wrong. The matcher recognizes this as an incorrect match, but one in which part of the information is correct — in particular, the matcher would recognize that Juan and Carlos occupy inappropriate slots in a representation that otherwise matches the prestored answer. This information allows the student to be informed that the action seems appropriate, but that the answer is wrong. The student is advised to reread the paragraph and to answer the question again.[4]

It should be noted that the lack of a more complex knowledge representation system underlying the matching makes for some limitations to this technique. For example, consider the following student response to the question as to what happened to Juan:

STUDENT> Carlos le dio unas puñaladas fatales a Juan.
(Carlos stabbed Juan fatally.)

In this case, the system would generate the following LCS:

(15) [$_{\text{Event}}$ CAUSE
([$_{\text{Thing}}$ CARLOS],
[$_{\text{Event}}$ GO$_{\text{Poss}}$
([$_{\text{Thing}}$ KNIFE-WOUND],
[$_{\text{Path}}$ TOWARD$_{\text{Poss}}$
([$_{\text{Position}}$ AT$_{\text{Poss}}$ ([$_{\text{Thing}}$ KNIFE-WOUND], [$_{\text{Thing}}$ JUAN])])])],
[$_{\text{Manner}}$ FATALLY])]

This would not match the LCS generated previously for "Juan murió." The system would need to infer that "Manner FATALLY" implies the death of the subject, but that is beyond the scope of the LCS system. We return to this issue in section 6.

In addition to question answering, the usefulness of the LCS framework and the matching algorithm extends to another lesson type provided by the foreign language tutor, i.e., that of Text Translation, in which the student must translate an English sentence into a sentence in either Arabic or Spanish. Again, the author provides a sample question (this time the sentence to be translated) and answer. The answer is parsed, analyzed and stored as described above. The student's translation and the prestored answer are matched, and the tutor is able to analyze whether the translation is correct or not.

4 LCS Editing Capability

During lexical-semantic analysis of a foreign language sentence, the intelligent tutor accesses a LCS dictionary for that language. Manual construction of this dictionary is a very tedious task; thus, we have designed and implemented a LCS Editor that makes the construction process much faster, easier, and less error-prone.

The heart of the LCS Editor is the LCS Entry Window shown in figure 5. This window is where LCS's are built and modified. During editing, the current state of the LCS is continuously updated and displayed in two windows, the "Tree Representation" window and the "Text Representation" window.

[4]It should be noted that the matcher cannot currently recognize that they are transposed, but

Figure 5: LCS Entry Window

At the top of the Editor window is a browser which allows the author to set the Type, Primitive, and Field of the selected (highlighted) node in the current LCS. The selected Type determines which Primitives are available, and the selected Primitive determines the Fields that are allowed. Some Primitives have no allowed Fields. The Editor only allows valid choices to be made.

The Editor also displays sample sentences for various combinations of Primitive and Field, to guide the author with respect to the meanings of the various terms. In figure 5, the words *We moved the statue from the park to the zoo* represent an example of a sentence using a word (i.e., *move*) that has the same Type, Primitive, and Field as the currently-selected node.

The LCS of this example contains nodes labeled X, Y, and Z. There are no Primitives by these names; rather, these nodes represent variables. Any nodes with the same variable name will eventually refer to the same item in the natural language sentence that this LCS is used to represent. For each Type, there is a Primitive called "var_*Type*." For example, under the Event type there is a primitive called "var_Event." If this were chosen as a Primitive, a window would appear asking for a variable name.

Under the browser is the Add Primitive button. Clicking this button allows the author to add a new primitive to the set that is currently defined, as well as specify which Type the primitive belongs to and which Fields are applicable.

The next item down is a group of controls labeled "LCS Characteristics." These are items that apply to the LCS entry as a whole. The author enters the word that the entry represents in the Text Field labeled "The Word." The Usage Comment button allows the author to enter a comment about the usage of this particular entry for documentation purposes. (It has no effect on the correctness of the entry.) Below this button is a popup menu labeled "English," reflecting the fact that this is the LCS entry for the English verb, *go*. Next to this is another popup menu labeled "Root," indicating that this LCS is for a word with root causality. If, for example, the LCS were representing the verb, *shove,* then this popup would have been set to "Causative."

Next to the group labeled "LCS Characteristics" is another group of controls labeled "Node Actions." These items apply solely to the node that is currently selected. In the current example, the root node of the tree is selected, so any of the "Node Actions" items that are chosen would apply to the root node. The selection of a node action would open up a secondary window (palette) that supports the annotation of LCS items with classifying markers (such as the :INT/:EXT markers given in example (8) earlier), semantic and syntactic feature information, and usage comments. If the tree grew large enough, vertical and/or horizontal scrollers would appear to allow the author to see different regions of the tree.

The LCS Editor is currently being used to enter vocabulary into the dictionaries for Spanish and Arabic.

---

simply that they are inappropriate names. Thus "Igor killed Fred" would also be judged similarly.

STUDENT>	Cuál es su apellido?
	(What is your name?)
TUTOR>	Ramirez.
STUDENT>	Su nombre?
	(Your first name?)
TUTOR>	Diego.
STUDENT>	Cuántos años tiene usted?
	(How old are you?)
TUTOR>	Yo tengo 24 años.
	(I am 24 years old.)
STUDENT>	Dónde fue la capturada?
	(Incorrect sentence — Where was the captured woman?)
TUTOR>	No comprendo.
	(I don't understand.)
STUDENT>	Dónde fue detenido?
	(Where were you captured?)
TUTOR>	Durante la redada yo estaba detrás la casa
	y salí corriendo.
	(During the raid, I was in back of the house and ran away.)
STUDENT>	Y entonces?
	(And then?)
TUTOR>	La policia me vio y fui arrestado.
	(The police saw me and I was arrested.)

Figure 6: Sample of Small Segment of Dialog with Simulated Spanish Speaker

5 Limited Domain Discourse

An area that is currently under investigation within the context of this research is the development of a tool which can incorporate the ability for the student to interact with a simulated native language speaker in a relatively unconstrained dialog. In one example of this mode of interaction the system might take the role of a person responding to questions, which the student is asking to the best of his or her ability. Figure 6 illustrates what might be a small segment of the dialog with the simulated Spanish speaker, "Diego Ramirez."

The ability to converse in such an unrestricted manner puts a significant burden on the natural language system. The system must be able to correctly analyze the questions (or statements) the student has produced and it must be able to generate a correct answer. This can require either a simple knowledge-based retrieval (as in the first three sentences of figure 6), or it can require the production of a more complex answer, depending on the question (as in the remaining sentences).

When the student's input cannot be analyzed (due to syntactic or semantic error), the system must produce an appropriate response that can keep the dialog going. For example, in the fourth sentence above, the student, who mistakes the inappropriate cognate "capturada" for the correct term "detenido" asks "Dónde fue la capturada?" (which translate roughly as "where was the captured woman?"). If we assume there is no captured woman the Spanish speaker knows about, he might

answer "No comprendo." — "I don't understand you." The student then finds the right word, asks the appropriate question, and the discourse continues.

Efforts to support this type of dialog interaction require the addition of a *planning component* with sophisticated semantic representations that can gauge the semantic appropriateness of student input and structure responses that move the dialog along in an appropriate fashion. These components are currently being added to the already mature syntactic analyzer. Also under implementation is a *generation component* that turns the output of the NLP interpretation into a response phrased appropriately in the foreign language. Figure 7 shows the planned design of the augmented system.

As an example, consider how the system would understand and produce the answer to one of the questions in the current dialog. The student types "Dónde fue detenido?" to the tutor, which passes this string to the syntactic parser. The syntactic parser produces a parse tree which recognizes this sentence as a "WH" clause (i.e., a question) with the appropriate syntactic relations among the words noted.

The parse tree is then sent to the lexical-semantic analyzer, which is responsible for producing a semantic structure which unambiguously captures the meaning of the sentence. This structure is created by combining the meanings of the separate words based on the rules of the language in question (encoded via appropriate parameterization) using the features prescribed in the lexicon for guidance. In this case, the various parts of the parse tree are used to produce the following LCS:[5]

(16) [$_{\text{Event}}$ CAUSE
([$_{\text{Thing}}$ X +human],
[$_{\text{Event}}$ GO$_{\text{Poss}}$
([$_{\text{Thing}}$ REFERENT +human +sg +p2],
[$_{\text{Path}}$ TO$_{\text{Poss}}$
([$_{\text{Position}}$ AT$_{\text{Poss}}$
([$_{\text{Thing}}$ REFERENT +human +sg +p2], [$_{\text{Location}}$ WH-LOCATION])])])],
[$_{\text{Manner}}$ FORCEFULLY])]

This roughly corresponds to the question "Where were you when you were caused to be forcefully transferred to someone else's possession?"

This LCS is then passed to the Discourse Planner which attempts to understand it in the context of the current dialog. Based on information about the situation, the system can infer that this is a question about location, that it concerns Diego Ramirez (the persona the system is trying to imitate), and that the question concerns his capture.[6]

Based on this analysis, the discourse planner is invoked. A number of planning operators may be applicable, each one chosen based on pattern matching against appropriate parts of the LCS. For example, questions about *where* something has occurred are handled by the following operator:

[5]Features have been included here for illustrative purposes. The actual LCS is even more complex; we omit some details not relevant to the current discussion.

[6]This is deduced via the use of a discrimination network which matches components of the LCS against target meanings in an efficient manner.

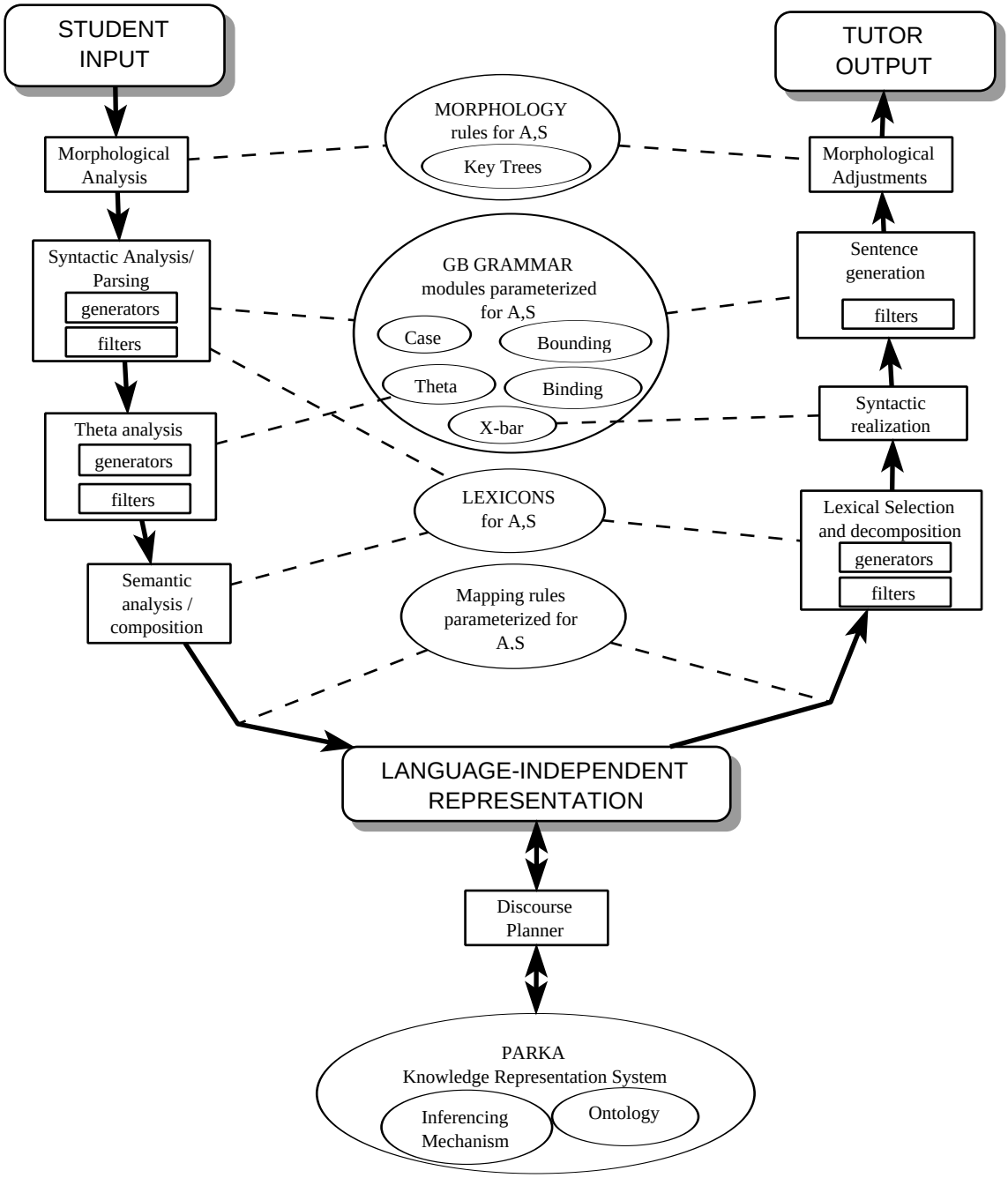

Figure 7: Diagram of Augmented Natural Language Processor for Tutor Project

(17) OPERATOR-SCHEMA *Where-interrogative*
PRECONDITION: FTYPE(Inst-Frame) ≡ WH
TYPE(Inst-Frame) ≡ Locative
BIND: Agent ← *Subject
Mode ← Process-clause[*Internal-Clause]
CONDITIONS:
Prev-Estab(Agent,?Location,Mode)
→ Reply(Already-desc)
Unknown(Agent,?Location,Mode)
→ Reply(Noncommittal,*where*)
OTHERWISE
Match-Structure([?Frame,episode = Mode,subj:= = Agent])

Roughly translated, this operator states that to answer questions about where something occurred, we need to check if we already have answered this question (if so, generate a reply to this effect). If not, but we don't know where this has occurred, then we need a non-committal response. Finally, if we do know where this has occurred, then we query for the LCS associated with the episode in question.

This query retrieves a set of frames corresponding to episodes in the knowledge base. The knowledge base is implemented using the PARKA frame language, a knowledge representation system developed at the University of Maryland (see Spector et al. (1990)). This language provides tools for the creation, editing, and querying of knowledge bases.[7] Associated with each frame is a corresponding LCS which can be used in producing the answer. Thus, for the example we are following, the answer would be found to be the following LCS:

(18) $[_{\text{Event}}\ \text{BE}_{\text{Loc}}$
$([_{\text{Thing}}\ \text{REFERENT}\ +\text{human}\ +\text{sg}\ +\text{p1}],$
$[_{\text{Position}}\ \text{AT}_{\text{Loc}}$
$([_{\text{Position}}\ \text{BEHIND}_{\text{Loc}}\ ([_{\text{Thing}}\ \text{REFERENT}\ +\text{human}\ +\text{sg}\ +\text{p1}],\ [_{\text{Location}}\ \text{HOUSE}])])],$
$[_{\text{Time}}\ \text{T1 overlaps T3}])]$

$[_{\text{Event}}\ \text{GO}_{\text{Loc}}$
$([_{\text{Thing}}\ \text{REFERENT}\ +\text{human}\ +\text{sg}\ +\text{p1}],$
$[_{\text{Path}}\ \text{AWAY-FROM}_{\text{Loc}}$
$([_{\text{Position}}\ \text{BEHIND}_{\text{Loc}}\ ([_{\text{Thing}}\ \text{REFERENT}\ +\text{human}\ +\text{sg}\ +\text{p1}],\ [_{\text{Location}}\ \text{HOUSE}])])],$
$[_{\text{Manner}}\ \text{RUNNINGLY}],$
$[_{\text{Time}}\ \text{T2} > \text{T1}])]$

Here T3 is the time of the raid in the current episode. These representations roughly correspond to a "a REFERENT was at a location which was behind the HOUSE. He ran from there to an unspecified location."

This LCS information is passed back to the lexical-semantic component which uses this information in the lexicon both to assign words to the variable entries ("REFERENT" and "HOUSE") as well as determining the syntactic structure that

[7] One of the most important aspects of PARKA is that it runs extremely efficiently on the massively-parallel Connection Machine. The work in this project uses only the serial implementation. For details about the parallel work (see Evett et al. (1993)).

best conveys the idea in the appropriate language. This is used to produce a syntactic structure which is transformed by the generator into the output response "Yo estaba detrás de la casa." A similar process is used to interpret the latter LCS and produce "Yo salí corriendo." Following this, time information is used to recognize that the running occurred at the same time as the raid, and thus "Durante la redada" is added to the set of sentences to be expressed. Finally a generational rule is used to combine the two into a more grammatical compound sentence and the final answer is produced:

TUTOR> Durante la redada yo estaba detrás la casa
y salí corriendo.
(During the raid, I was in back of the house and ran away)

The discourse planner is also invoked when the unification algorithm fails to find a query that matches the LCS produced. For example, following the sentence above, the student might type:

STUDENT> Y entonces?
(And then)

In this case, no predicted query matches, and a default rule in the discourse planner is used. To handle such cases, the planner tracks the dialog, keeping account of the context of the discourse. Since the last answer produced focused on running away, this sentence is requesting information about the next temporal event (signaled by "entonces"), and the knowledge base is checked to see what the next event was. Again, an appropriate response such as:

TUTOR> La policia me vio y fui arrestado
(The police saw me and I was arrested)

would be produced.

The use of a discourse planner to track the dialog also allows other behaviors to be introduced via similar mechanisms. Responses such as "I told you that already," "I don't know why you want to know that," and "I won't answer that question" can easily be produced via this mechanism.

The examples given so far have assumed a full immersion mode, where the feedback is as potentially unhelpful as a real detainee. However, modes with lower immersion level can also be envisioned. So, for example, in order to diagnose the fact that "Dónde fue la capturada" (where was she captured) is incorrect, the discourse model has to have reported that the discourse doesn't mention any females being captured. The system might use this information to generate a message telling the student why the interogatee failed to understand.

Other possible errors involve lexical semantic or syntactic errors. For example, the student might make an error of the following type, using the wrong form of "be":

TUTOR> Era detrás de la casa y salí corriendo
(I took place / am equal to the back of the house and ran away)

This is for the intended form: "I was in back of the house and ran away."

We expect to be able to configure the system to simply ignore the error or to tell the student that he or she has used the wrong form of "be" by pointing to the part of the string where the error has occurred. For selected errors of high priority or

high likelihood of occurrence, students can get a grammar lesson (e.g. on the senses of "be") from a pull-down window. The tutor also keeps track of numbers of errors and can direct the student to exercises of the appropriate style (determined by its tutoring rules) to remediate particular grammatical problems.

More sophisticated semantic representations that can take information from a knowledge base for the purpose of responding to queries is crucial to run a dialog. In particular, two main types of information must be encoded: information about discourse processing and information about the domain of discourse. Although how to design a formal model of discourse processes is still largely an open question, a number of approaches have shown to be useful when the domain of discourse is limited and controlled.

6 Conclusions

This chapter described the use of an LCS-based semantics for question-answering exercises in foreign language training. We started by reviewing the LCS model and we showed how this representation can be used to support a question answering lesson as well as a limited domain discourse. An authoring tool for entering the relevant semantic knowledge was also described.

An area of current investigation beyond the work described here is that of defining the relation between the lexical-semantic representation and the knowledge representation. Our goal is to understand a sentence where the "message" contains a spatial relation — in particular, where the sentence conveys information about the location or path of physical entities in the real, physical world. For example, if we consider the status of the sentence *Daniel drove to the south of Colorado*, we quickly determine that the phrase *the south of Colorado* is ambiguous. One interpretation of this sentence is that Daniel drove to southern Colorado. That is, the phrase *the south of Colorado* refers to the region inside of Colorado that is considered its south. The lexical-semantic structure for this part-to-whole relation, where the "part" is the meaning of the entire phrase and the "whole" is Colorado, is viewed as a "place-place" relation in the knowledge representation. Thus, the knowledge representation is used to check for a part-whole interpretation when it encounters two "place" predicates in a lexical-semantic representation.

Another future direction is the exploration of an extension of this linking of the semantics and a knowledge representation system to handle problems such as the "stabbed fatally" case referred to in section 3. It is clear that to be able to fully analyze a student's answer, we must have an inferencing system capable of "understanding" both the paragraph shown to the student and the students' response to the query. Providing such a capability on a "lesson by lesson" basis is within the capability of current AI techniques; in essence, it would require techniques similar to those described in the discourse lesson (section 5). Unfortunately, this is still a labor intensive process, and until large shared "common sense" knowledge bases become available, making such a system authorable remains a major research challenge.[8]

[8]Efforts in this direction include the US Knowledge Sharing Effort and the Japanese project "NOAH" which both aim to provide tools for building and sharing very large knowledge bases.

Acknowledgements

Dr. Hendler is supported in part by grants from NSF (IRI-8907890), ONR (N00014-J-91-1451), AFOSR (F49620-93-1-0065), the ARPA/Rome Laboratory Planning Initiative (F30602-93-C-0039) and by ARI (MDA-903-92-R-0035, subcontract through Microelectronics and Design, Inc.) Dr. Hendler is also affiliated with the UM Institute for Systems Research (NSF Grant NSFD CDR-88003012). Dr. Dorr is supported in part by grants from NSF (IRI-9120788), ONR (N00014-92-J-1929), ARO (DAAL03-91-C-0034, subcontract through Batelle Corporation), and by ARI (MDA-903-92-R-0035, subcontract through Microelectronics and Design, Inc.).

References

1. Chomsky, Noam A. (1981) *Lectures on Government and Binding*, Foris Publications, Dordrecht, Holland.

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

3. Dorr, Bonnie J. (1992) "Lexical Semantics for Interlingual Machine Translation," *Machine Translation*, 7:3.

4. Dorr, Bonnie J. (1993a) "Interlingual Machine Translation: A Parameterized Approach," *Artificial Intelligence*, 63:1&2.

5. Dorr, Bonnie J. (1993b) *Machine Translation: A View from the Lexicon*, MIT Press, Cambridge, MA.

6. Evett, Matt, James Hendler, and Lee Spector (1993) "Parallel Knowledge Representation on the Connection Machine" *International Journal of Parallel and Distributed Computing,* in press.

7. Grimshaw, Jane (1990) *Argument Structure*, MIT Press, Cambridge, MA.

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

9. Hale, Kenneth and S. Jay Keyser (1986b) "A View from the Middle," Lexicon Project Working Paper 10, Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA.

10. Hale, Kenneth and S. Jay Keyser (1989) "On Some Syntactic Rules in the Lexicon," manuscript, Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA.

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

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

13. Jackendoff, Ray S. (1990) *Semantics Structures*, MIT Press, Cambridge, MA.

14. Levin, Beth (1985) "Lexical Semantics in Review," Lexicon Project Working Paper 1, Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA.

15. Levin, Beth (1993) *English Verb Classes and Alternations: A Preliminary Investigation*, University of Chicago Press, Chicago, IL.

16. Levin, Beth and Malka Rappaport (1986) "The Formation of Adjectival Passives," *Linguistic Inquiry*, 17, pp. 623–662.

17. Mel'čuk, Igor and Alain Polguère (1987) "A Formal Lexicon in Meaning-Text Theory (Or How to Do Lexica with Words)," *Computational Linguistics*, pp. 261–275, 13:3–4.

18. Olsen, Mari Broman (1991) "Lexical Semantics, Machine Translation, and Talmy's Model of Motion Verbs," Linguistics Working Paper, Volume 3, Northwestern University, Evanston, IL.

19. Pustejovsky, James (1988) "The Geometry of Events," Lexicon Project Working Paper 24, Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA.

20. Pustejovsky, James (1989) "The Semantic Representation of Lexical Knowledge," in *Proceedings of the First International Lexical Acquisition Workshop*, IJCAI-89, Detroit, MI.

21. Pustejovsky, James (1991) "The Syntax of Event Structure," *Cognition*, 41.

22. Rappaport, Malka and Beth Levin (1988) "What to Do with Theta-Roles," in Wendy Wilkins (ed.), *Thematic Relations*, Academic Press.

23. Rappaport, Malka, Mary Laughren, and Beth Levin (1987) "Levels of Lexical Representation," Lexicon Project Working Paper 20, Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA.

24. Sams, Michelle R. (1993) "An Intelligent Foreign Language Tutor Incorporating Natural Language Processing," *Proceedings of Conference on Intelligent Computer-Aided Training and Virtual Environment Technology,* NASA: Houston, TX.

25. Schank, Roger C. (1972) "Conceptual Dependency: A Theory of Natural Language Understanding," *Cognitive Psychology*, 3, pp. 552–631

26. Schank, Roger C. (1973) "Identification of Conceptualizations Underlying Natural Language," in Roger C. Schank and K. M. Colby, *Computer Models of Thought and Language* Freeman, San Francisco, CA, pp. 187–247.

27. Schank, Roger C. (ed.) (1975) *Conceptual Information Processing*, Elsevier Science Publishers, Amsterdam, Holland.

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

29. Sharp, Randall M. (1985) "A Model of Grammar Based on Principles of Government and Binding," Master of Science thesis, Computer Science, University of British Columbia.

30. Spector, Lee, James Hendler, and Matt Evett (1990) "PARKA: Parallel Knowledge Representation on the Connection Machine," UMIACS TR 90-22, CS Dept. TR 2409.

31. Vauquois Bernard, and Christian Boitet (1985) "Automated Translation at Grenoble University," *Computational Linguistics*, 11:1, pp. 28–36.

32. Zubizarreta, Maria Luisa (1982) "On the Relationship of the Lexicon to Syntax," Ph.D. thesis, Department of Linguistics and Philosophy, Massachusetts Institute of Technology, Cambridge, MA.

33. Zubizarreta, Maria Luisa (1987) *Levels of Representation in the Lexicon and in the Syntax*, Foris Publications, Dordrecht, Holland/Cinnaminson, USA.