Facilitating the Rapid Development of Language Understanding Interfaces for Tutoring Systems

Carolyn Penstein Rosé
LRDC, University of Pittsburgh,
Pittsburgh, PA 15260
rosecp@pitt.edu

Introduction

A variety of studies in recent years have focused on assessing the relative effectiveness of different *human* tutorial strategies and investigating the role of language interaction in such strategies, including (Chi *et al.* 1989) (Chi *et al.* 1994) (VanLehn, Jones, & Chi 1992) (VanLehn, Siler, & Baggett 1998) (Rosé *et al.* 2000). These studies leave open the questions of whether the same tutoring strategies that are effective in human tutoring can be emulated in ITSs with similar effectiveness, and whether this emulation can in fact be achieved in a cost effective way. Without answers to these questions, the direct relevance of such findings about human tutorial dialogue to the field of intelligent tutoring is called into question. While interest in dialogue interfaces for tutoring systems is rapidly growing, real progress in this direction has been greatly hindered by the tremendous time, effort, and expertise that is required to construct such interfaces. This paper describes a design for and current progress towards building DAIENU[1], the Domain knowledge Authoring Interface for Extractive Natural language Understanding. The purpose of the DAIENU tool set is to facilitate the rapid development of robust and efficient natural language understanding interfaces for tutoring systems.

The DAIENU tool set is part of the larger Knowledge Construction Dialogue Authoring Tool Suite developed in the context of the Atlas project to automate the authoring of all domain specific knowledge sources required to build a domain specific dialogue-based tutoring system using the general Atlas architecture (Freedman *et al.* 2000). In particular the current prototype version of the tool suite automates the construction of recipes for the APE tutorial dialogue planner (Freedman 2000) and semantic rules for the LCFlex robust parser (Rosé and Lavie, to appear). The prototype KCD Authoring Tool Suite is currently operational but continuing to be developed and refined. This prototype tool suite was recently used to develop knowledge sources for implementing directed lines of reasoning targeting 50 physics rules covering all aspects of Newtonian mechanics. The entire authoring process, including initial authoring, review by collaborating physicists, and revision required only two man months of development time. The result was a running dialogue system currently being pilot tested with naive human subjects. The running directed lines of reasoning will eventually replace the teaching hints previously used in the Andes tutoring system (VanLehn *et al.* 2000).

The KCD Authoring tool suite is meant to allow authors with little or no computational linguistics expertise to rapidly develop dialogue interfaces for their applications. Thus, the goal for the authoring interface is to insulate the author from directly addressing the underlying computational linguistic aspects of dialogue planning and language understanding. The author is free to focus fully on pedagogical issues, such as what directed lines of reasoning are most effective for teaching particular concepts, how misconceptions and missing knowledge manifest themselves in student answers, and which aspects of student answers are relevant for evaluating their content. Furthermore, the interface is flexible enough to allow the author to be involved in the computational linguistic aspects of development if the author so chooses. For example, advanced authors may consider system integration issues such as what tokens or logical forms should be passed on to the dialogue manager or back end tutoring system. Thus, it is adaptable to differing levels of expertise on the part of the author.

The Authoring Process

The DAIENU tool set is meant to facilitate the rapid development of domain specific sentence level language understanding interfaces that can extract desired information from student input. The core of the target domain specific system is the domain independent CARMEL core understanding component (Rosé 2000a). CARMEL makes use of two primary domain independent knowledge sources, namely a broad coverage syntactic parsing grammar and a large scale lexicon than can easily be augmented with domain specific vocabulary and idiomatic construction entries. It provides a meaning representation formalism for specifying a domain specific meaning representation declaratively.

[1] Daienu is a Hebrew word pronounced die-ay-noo. It means "enough for us".

Thus, applying CARMEL to a new domain requires designing a meaning representation, entering domain specific vocabulary and relevant idiomatic constructions into the lexicon, and finally linking the semantic constructor functions into the lexicon. Currently the process of applying CARMEL to a new domain is done by hand; however, as the DAIENU tool set is further developed, more and more of this work will be automated.

The DAIENU authoring process takes place after an initial pass with the KCD Authoring Tool Suite's GUI interface. The KCD Authoring GUI provides templates that allow authors to develop relatively high level specifications for directed lines of reasoning. Recipes are the building blocks out of which the high level specs are built. Primitive recipes encode the most basic steps executed by a tutorial planner. A Primitive recipe may simply encode a tutor explanation or a question for eliciting a particular piece of information. More abstract recipes are composed of one or more steps that may themselves be abstract recipes or primitive recipes. Each recipe is associated with a tutorial goal. In this way, the internal representation may encode alternative ways of achieving the same tutorial goal.

Each Primitive recipe that encodes a question used to elicit a piece of information from a student is associated with a hierarchical answer representation. The answer representation is intended to allow for a wide range of types of answers and some variation in how answers are expressed by students. Answers are divided into parts. For example, if the tutor asks, "Can you describe a reaction force in relationship to its associated action force?" The expected answer has three parts: Same magnitude, opposite direction, same type. These three parts may appear in any order without affecting the correctness of the answer. Also, each part is independent of the others in that a student may get one or more parts correct while missing the other. Whether a student gets the direction of the reaction force correct is independent of whether the student realizes that the magnitude is the same, for example. By dividing the expected answer into three parts, we avoid the need to list every permutation of the three parts.

For each answer part, we list a number of alternatives. For example, the student may have said same direction, or opposite direction, or left out the direction altogether, or gave an explicit direction like left or right. For completeness, we always include <anything else> as one of the alternatives, which covers the case where the associated part of the answer was left out of what the student said altogether. Each alternative in the list has associated with it a status, such as expected, partial, wrong, etc. Every alternative that is not correct has associated with it a list of tutorial goals that must be achieved by the tutorial planner in order to remediate that wrong answer. A simple extension allows for alternative answers that are not composed of the same number of parts. Alternative answer parts are entered by authors in text form.

The first stage in DAIENU's authoring process is to assemble a list of concept labels paired with a set of example texts. Out of the complete set of answer parts entered by authors during the initial pass, there may be many answer parts that are equivalent in meaning that may or may not have been expressed in exactly the same words. DAIENU first guides authors in clustering together answer parts with equivalent meaning. It does this by suggesting answer parts that share content words and providing a search interface for allowing authors to quickly identify other candidates from the complete set. During and after the clustering process, authors may enter alternative phrasings of the same concept either by constructing examples or by finding examples in a corpus.

A machine learning stage described below then induces patterns for the concept labels to be used by the target system for matching against student input. An interactive refinement stage allows authors to influence the resulting induced patterns. During this interaction-refinement process, the author is provided with a set of novel texts, produced and labeled by the system. The author is then required to indicate which elements of the set are correctly annotated and which are not. DAIENU then uses this information to further develop its induced patterns.

Note that authors are not constrained to proceed through the authoring stages in a linear fashion. Previous stages can be revisited in order to allow for continued refinement at all levels.

Learning Algorithms for Constructing Domain Specific Knowledge Sources

The result of the authoring process described above is a set of abstract conceptual classes such that each class pairs a concept label or logical form with a set of example natural language expressions that have each been analyzed using the CARMEL core understanding component. DAIENU's task is then to find commonalities between the analyses within each set in order to construct the most compact set of general patterns that can match against the representations for each example in the set. The purpose is to find the most general conditions under which a novel expression can be labeled with the corresponding concept label or logical form. These general patterns are what the target system uses to match against student language input. To perform the generalization task we plan to use techniques similar to those being used for automatic template generation in the information extraction and text classification communities (Glickman & Jones 1999) (Bruninghaus & Ashley 1999) (Riloff & Shephard 1997) (Riloff 1996).

Consider the following examples from the Newtonian Physics domain that each express the idea that a student drew a vector and that the resulting vector points down.

1. The student drew the vector to point down.
2. The vector was drawn by the student to point down.

3. I saw the vector that the student drew to point down.
4. The vector that was drawn by the student pointed down.
5. I saw the vector that was drawn to point down by the student.
6. The vector that the student drew slanted down.
7. The vector that was sketched by the student slanted down.
8. The vector drawn by the student was oriented downwards.
9. The student's vector pointed down.

Despite the difference between the surface realizations of these sentences, they have aspects in common on the level of their respective deep functional analyses. In examples 1 through 5 "The student" is the deep subject of "to draw", and "the vector" is the deep object of "to draw" as well as the deep subject of "to point". In example 6, the verb "to slant" is used in place of "to point". Example 7 replaces "to point" with "to slant" and replaces "to draw" with "to sketch". Finally, example 8 replaces "to point" with "to be oriented". Nevertheless, the substitutions made in examples 6 through 8 are on the level of rough synonyms. Example 9 contains the proposition "The vector points down", as in the previous examples, but uses the possessive to indicate that the student drew the vector. Thus, its functional analysis is distinct from the preceding examples.

Notice that it is not sufficient to take the intersection of all of the features found in the analyses of the set of provided examples in order to find the most general set of features that covers the appropriate range of student language expressions. Specifically, see how Example 9 does not contain the same information as the preceding examples encoded in the same way. Ideally the learning process should cluster the first 8 examples together since a single abstract pattern could cover the variation found within that set quite easily. The 9th example would then be handled separately. Syntactic analyses can be clustered into groups using an unsupervised clustering algorithm that compares analyses in terms of which features they share. In order to facilitate domain dependent generalizations, such as clustering "point", "slant", and "orient" together, an automatically constructed rough domain specific ontology created by means of LSA could be used to identify clusters of similar words. Alternatively, a lexical resource such as WordNet could be used a similar way. Using this rough ontology, specific lexical heads could be replaced by tokens representing more abstract domain specific concept. This ontology makes it possible to make such domain specific abstractions, however it is not by itself sufficient to determine which level of abstraction is appropriate for particular concept labels. It does, however, provide a set of alternative hypotheses on progressively more abstract levels of representation.

The task remains then of selecting the set of features that describe the weakest conditions under which a meaning representation structure can always be labeled with the associated concept label. One possible solution would be to use a decision tree learning algorithm to identify the most appropriate set of features from a large set of possible alternatives (Quinlan 1990) (Kakes & Morris 1996) (Helmbold & Schapire 1997) (Martin 1997) (Utgoff, Berkman, & Clouse 1997) and then using interaction to select between alternative learned trees. Testing hypothesized trees could be accomplished by generating text corresponding to alternative feature structures that match the conditions represented by the hypothesized tree. The user of the authoring system would then be asked to mark the generated examples that are not classified correctly. Marking any of the examples indicates that the hypothesis is not specific enough. Similar forms of user interaction for the purpose of hypothesis confirmation have played a large role in our previous research on interactive recovery from parser failure (Rosé & Levin 1998) (Rosé 1997) (Rosé & Waibel 1997).

Because the learned patterns are based on functional roles assigned by CARMEL's syntactic parsing grammar, it should be possible to automatically link learned patterns into the lexicon. Performing this task only requires matching lexical items to meaning representation types and functional roles to semantic roles. Our current research focuses on developing these learning aspects of the proposed tool set. Note that the target domain specific knowledge sources will be encoded in the same formalism used now to apply CARMEL to new domains by hand. Thus, expert authors would have the ability to fine tune or change the learned knowledge by hand by modifying the automatically constructed knowledge sources.

The CARMEL Core Understanding Component

The CARMEL core understanding component (Rosé 2000a; 2000b), is the heart of the DAIENU tool set. It provides the underlying linguistic knowledge that makes it possible to generalize the examples annotated by the authors into patterns that can match against unseen data, and thus be used in an interface for processing student input.

CARMEL in the Context of Previous Work

The majority of current dialogue based tutoring systems employ shallow approaches to language understanding. For example, the AutoTutor system (Wiemer-Hastings *et al.* 1998) uses a purely empirical approach to interpretation, namely Latent Semantic Analysis (LSA). Although "bag of words" approaches such as LSA (Landauer, Foltz, & Laham 1998) (Laham 1997) and HAL (Burgess, Livesay, & Lund 1998) (Burgess & Lund 1997) have enjoyed a great deal of popularity and success in some domains, they miss key aspects of meaning communicated structurally through scope and subordination such as the proper interpre-

tation of negation and causality. Therefore, they miss such distinctions as the difference between "the force of the table pushing against the book" and "the force of the book pushing against the table". These aspects of natural language have been well studied in the field of linguistics and well addressed within rule-based approaches to language understanding. Current dialogue based tutoring systems with rule based language understanding interfaces include CIRCSIM-TUTOR (Glass 1999), BE&E tutor (Rosé, Di Eugenio, & Moore 1999), and Atlas-Andes (Freedman et al., 2000) . A great deal of computational linguistics research has focused on the development of robust domain independent algorithms for rule-based language understanding. Nevertheless, a significant knowledge engineering effort is required to develop domain specific knowledge sources to apply these algorithms to specific tasks. Broad coverage domain specific knowledge sources for non-trivial applications commonly take two or more man years to develop by experienced computational linguists. The goal of the tool set described in this paper is to dramatically reduce these requirements, making it practical for researchers who may lack expertise in computational linguistics to quickly construct sentence level language understanding interfaces for their tutoring applications.

The approach to language understanding embodied in the DAIENU tool set is called extractive language understanding (ELU) because of its similarity with information extraction (IE) approaches, such as (Glass 1999). The CARMEL core understanding component (Rosé 2000a; 2000b) provides deep syntactic functional analyses from which DAIENU induces abstract patterns that are then used to match against analyses of novel student input. The primary difference between ELU and typical IE approaches is that ELU employs a deeper level of syntactic analysis. While typical IE approaches limit their syntactic analysis to surface syntax, ELU uncovers deep syntactic functional relationships obscured in surface syntax by extraction and passivization. This deep syntactic functional analysis allows for a greater degree of generalization across paraphrases of the same idea, such as between "I drew the vector to point down." and "The vector was drawn pointing down." This greater generalization power is advantageous in the context of an authoring environment because it means that many fewer examples are needed for the purpose of inducing effective patterns for matching against student input.

Robustness at Each Level of Processing

Input understanding in CARMEL proceeds in three distinct stages. See Figure 1. The lexical preprocessing stage, which is the initial stage, is responsible for performing morphological analysis on each word in the input sentence and retrieving all lexical entries that match the root form of each word from the lexicon. Subsequently, the parsing stage is responsible for performing a syntactic and semantic analysis with these lexical entries in so far as the parser's flexibility settings will allow. When necessary, a repair stage is responsible for assembling the pieces of a fragmentary parse when no complete parse is possible.

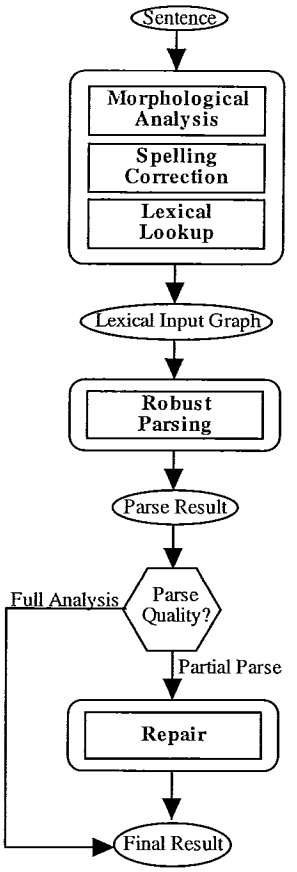

Figure 1: **CARMEL architectural overview**

The key to robustness in the lexical preprocessing stage is the coordination of morphological analysis, lexical lookup, and spelling correction. For each word, the morphological analyzer first constructs a list of possible segmentations of it into morpheme. For each possible segmentation, the associated lexical entries are retrieved from the lexicon. If none of the root forms from any of the possible segmentations correspond to an entry in the lexicon, a list of possible replacement words are obtained from the spelling corrector (Elmi & Evens 1998) based on the form of the word, the previous word and the next word. Based on its internal heuristics, the spelling corrector chooses either to split the word, combine the word with a neighboring word, or replace with whole word with one or more alternative words.

The key to robustness in the parsing stage is the LCFLEX robust parser (Rosé and Lavie, to appear) (Lavie & Rosé 2000). The LCFLEX parser robustly applies CARMEL's broad coverage syntactic parsing grammar by introducing various types of flexibility only as needed. Its repertoire of parameterized flexibility features includes skipping over words, insertion of

words or non-terminal categories in specific contexts, and relaxation of grammatical constraints expressed via feature unification. Because broad coverage syntactic grammars naturally produce a great deal of ambiguity, LCFLEX's ambiguity packing and pruning strategy as well as its statistical disambiguation make it an ideal choice for our application. When the flexibility allowed at parse time is not sufficient to construct an analysis of a sentence deviating too far from the grammar's coverage, a fragmentary analysis is passed on to the repair module, which quickly assembles the fragments.

CARMEL is fully implemented and has been evaluated as a whole and in part in a number of previous publications. In a small evaluation involving 100 spontaneous student utterances ranging between 1 ans 20 words long in the Newtonian Physics domain, CARMEL was able to achieve 87% coverage with an average run time requirement of .1 second per utterance (Freedman *et al.* 2000). CARMEL's LCFLEX robust parser has been tested extensively in the context of the JANUS large scale multi-lingual speech-to-speech translation system, achieving 73.3% acceptable translations in a recent evaluation over a set of 500 randomly selected spontaneous utterances in the Appointment Scheduling domain (Rosé and Lavie, to appear). In a similar evaluation over a separate set of 300 sentences in the same domain, LCFLEX achieved 75.0% acceptable translations while LCFLEX paired with CARMEL's AUTOSEM repair algorithm achieved a total of 78.7% acceptable translations (Rosé 2000a).

The Meaning Representation Specification

The meaning representation structures constructed by CARMEL can be used in intelligent tutoring systems to identify commonalities in propositional content between student answers that express the same conceptual knowledge but encode it differently. The meaning representation specification defines the frame based language that is used for this purpose. In particular, it defines the set of semantic types that together specify the set of frames and atomic feature values that make up the domain specific frame-based language, which slots are associated with each frame, and what range of frames and atomic feature values may fill each of those slots. When the meaning representation specification is compiled, constructor functions are created corresponding to each semantic type. These semantic constructor functions are first used at parse time to build up semantic representations in parallel with syntactic ones as in the Glue Language Semantics approach (Dalrymple 1999) (Dalrymple, Lamping, & Saraswat 1993). These same constructor functions are then used in a repair stage to compose the fragments returned by the parser in the cases where the parser is not able to obtain a complete analysis for an extra-grammatical input sentence.

CARMEL provides a simple formalism for defining meaning representations. Some sample entries are displayed in figure 2. Each entry corresponds to a semantic type and contains five fields: `:type`, `:isa`, `:instances`, `:vars`, and `:spec`. Some sample entries for the Newtonian physics domain are displayed in Figure 2. The `:type` field simply contains the name of the type. The `:vars` field contains a list of variables, each corresponding to a semantic role. The `:spec` field associates a frame and set of slots with a type. For each slot, the `:spec` field contains the name of the slot, the most general type restriction on the slot, and a specification of where the slot filler comes from. This third piece of information can be either a variable name, indicating that whatever is bound to that variable is what should fill that slot, or a function call to another semantic constructor function, allowing types to specify constraints at more than one level of embedding. Similar to the `:spec` field, the `:instances` field associates a list of atomic values with a type.

Inheritance relations are defined via the `:isa` field. Types inherit the values of each subsuming type's `:instances`, `:vars`, and `:spec` fields. Further details about the inheritance and compilation algorithms are deleted because of space restrictions.

The Lexicon

```
(:morph draw
 :syntax
   ((cat vlex) (root draw) (vform bare)
    (irreg-past +) (irreg-pastpart +)
    (features vveryvingpast)
    (proot (*or* near on upon to at from))
    (subcat (*or* pp np-advp np-to-inf-oc
       np-for-np intrans np np-as-np advp))
    (semtag draw1))
 :semantics (draw1 <draw> (subject agent)
             (negation polarity)
             (object theme)))

(:construct (<clause>)
 :syntax ((cat clause) (semtag know1))
 :constraints (((x1 root) = (*or* have get))
               ((x1 object root) =
                          (*or* clue idea))
               (x0 = x1)
               ((x0 root) <= know)
               ((x0 object) = *remove*)))
```

Figure 3: **Sample Lexical Entries**

CARMEL is a lexicon driven approach to semantic interpretation. As in the Glue Language Semantics approach (Dalrymple 1999) (Dalrymple, Lamping, & Saraswat 1993), the lexicon serves as a clean interface between syntactic and semantic knowledge. The CARMEL lexicon is built on top of the large scale COMLEX lexicon (Grishman, Macleod, & Meyers 1994) available from the Linguistic Data Consortium. The CARMEL lexicon contains both regular lexical entries for individual words as well as construct entries for idiomatic and otherwise non-compositional expressions.

```
(:type <*event>
 :isa (<>)
 :instances nil
 :vars (agent time polarity)
 :spec ((polarity [+/-] polarity)
        (agent <*who> agent)
        (time <*time> time)))

(:type <draw>
 :isa (<*event>)
 :instances nil
 :vars (theme)
 :spec ((frame *draw)
        (theme <*thing> theme)))

(:type [+/-]
 :isa (<>)
 :instances (+ -)
 :vars nil
 :spec nil)

(:type <magnitude>
 :isa (<*thing>)
 :instances nil
 :vars (ref-object)
 :spec ((frame *magnitude)
        (ref-object <vector> ref-object)))

(:type <velocity-vector-mag>
 :isa (<magnitude>)
 :instances nil
 :vars (timept quantity orientation object)
 :spec ((ref-object <velocity-vector>
          (<velocity-vector> timept quantity
            orientation object))))
```

Figure 2: **Sample meaning representation entries for the Newtonian Physics domain**

In general, we have taken a compositional approach to semantic interpretation, building up the meaning of an expression from the meanings of its constituent parts. However, idiomatic expressions are not correctly interpreted compositionally. Thus, these non-compositional expressions are represented as construct entries so that their meaning can be associated with the whole expression as a unit.

Regular lexical entries contain three fields: `:morph`, `:syntax`, and `semantics`. The `:morph` and `:syntax` fields are taken from COMLEX. The `:morph` field contains the root form of the word. The `:syntax` field contains the list of features assigned to that root form in the COMLEX lexicon. Often the `:syntax` field contains a disjunction of feature structures wherever there are multiple entries associated with a single root form in COMLEX. For example, COMLEX contains seven entries for "draw", although only one is displayed in Figure 3. To the original `:syntax` field, we have added a `semtag` feature. Semtag feature values correspond to mappings between syntactic roles and the arguments of semantic constructor functions. The `:semantics` field defines the specific mappings associated with each `semtag` feature value. Each mapping needs only be defined the first time the corresponding semtag value is used in a lexical entry.

Construct entries are used for non-compositional idiomatic expressions as well as bound collocation (Cruse 1986), which are compositional although they display idiom-like behavior. Idioms can be seen as sharing properties both with words and with phrases. To accommodate this phenomenon, construct entries can be seen as blurring the distinction between lexical entries and grammar rules. Our formalism for expressing construct entries is equivalent in power to the formalism for representing idiomatic expressions in (van den Berg, Bod, & Scha 1994). Construction entries contain a `:construct` field to replace the `:morph` field. It can contain a list of one or more words, grammar symbols, or a combination of the two. Thus, construct entries can leverage off of syntactic generalizations from the grammar and easily specify how the corresponding expression should be inflected. Construct entries contain a constraints field wherein constraints can be placed on the feature structures dominated by the grammar symbols found in the construct field. For example, the construct entry in Figure 3 represents the idiom "to have a clue". It matches against clauses. If the verb and direct object match the associated constraints, the verb root is changed to "know" and the direct object is deleted. Our design principle is one idiom, one rule. Our construct entry design makes it possible with one rule to match the great deal of variation allowed by the "have a clue" idiom: "I don't have a clue in the world", "I don't have any idea", "I haven't got the foggiest clue", "Get a clue!", "Haven't you any clue?", "I have got a clue." At the same time, it does not misfire on expressions that look similar but do not mean the same thing such as "I have got to clue you in."

Broad Coverage English Parsing Grammar

The underlying principle behind the CARMEL approach to syntactic analysis is to provide a solid foundation for semantic interpretation. We have incorporated aspects of both Functional Grammar (Halliday 1985) and Lexical Functional Grammar (Bresnan 1982) in our approach. Our usage of Functional Grammar has been primarily limited to our analysis of tense.

As mentioned, the focus of our approach has been on constructing deep syntactic functional analyses. Deep functional syntactic representations have been demonstrated to translate directly into quasi logical formulas (van Genabith & Crouch 1996) and thus provide useful input for semantic interpretation (Halvorsen 1997) (Dalrymple 1999)
(Dalrymple, Lamping, & Saraswat 1993).

The underlying grammar formalism used by LCFLEX is a unification-augmented context-free grammar formalism that was originally developed for the Generalized LR Parser/Compiler (Tomita 1990) (Tomita 1987) at the Center for Machine Translation at Carnegie Mellon University. As an extension to LCFLEX's pseudo-unification grammar formalism, AUTOSEM provides the `insert-role` function as an interface to allow semantic interpretation to operate in parallel with syntactic interpretation at parse time. When the `insert-role` function is used to insert a child constituent into the slot corresponding to its syntactic functional role in a parent constituent, the child constituent's semantic representation is passed in to the parent constituent's semantic constructor function as in the Glue Language Semantics approach to interpretation (Dalrymple 1999). AUTOSEM's lexicon formalism allows semantic constructor functions to be linked into lexical entries by means of the `semtag` feature. Each `semtag` feature value corresponds to a semantic constructor function and mappings between syntactic functional roles such as `subject`, `direct object`, and `indirect object` and semantic roles such as `agent`, `activity`, or `time`. See Figures 2 and 3 for example meaning representation definitions and lexical entries respectively.

The CARMEL Interpretation Process

Consider the example sentence "Should the vector have been drawn to point down?" and its parsed result displayed in Figure 4. This sentence expresses two propositions, one about drawing a vector and one about the vector pointing down. The verb `draw` as displayed in Figure 3, has as one of its subcategorization tags `np-to-inf-oc`, which indicates that the verb can take a noun phrase direct object that acts also as the subject of a to-infinitival clausal argument. In this case, the verb `draw` is passivized, so its surface syntactic subject is its deep syntactic object. Thus, "the vector" is both the deep object of "draw" as well as the deep subject of "point". When a verb, such as `draw`, appears in its past participle form following a form of the verb `to be`, an analysis is constructed for `draw` with the `passive` feature assigned the value `+`. When a passivized verb with subcategorization tag `np-to-inf-oc` appears with only a to-infinitival clause as its argument, as in "drawn to point down", one analysis the grammar produces is a structure with a representation of "point down" as a modifying clause, and features that indicate that the surface syntactic subject should be assigned two roles, namely, deep object and subject of the modifying clause.

Sentence:
Should the vector have been drawn to point down?

```
((mood *yes-no-interrogative)
 (root draw)
 (tense past)
 (modal should)
 (negation -)
 (passive +)
 (object
     ((root vector)
      (agr 3s)
      (definite +)
      (adjunct
          ((root point)
           (tense infinitive)
           (negation -)
           (passive -)
           (subject
               ((root vector)
                (agr 3s)
                (definite +)))
           (modifier ((root down)))))))
(semantics
    (*multiple*
        ((frame *orient)
         (theme ((frame *vector)))
         (direction ((frame *down))))
        ((frame *draw)
         (theme ((frame *vector)))))))
```

Figure 4: **Sample Parser Output**

Each time a constituent is assigned a functional role in relationship to `draw`, the constructor function for the semantic type <draw> is called. The semtag `draw1` indicates how the syntactic functional roles assigned by the grammar correspond to argument positions in the constructor function for <draw>. Thus, when the constructor function is called, the corresponding argument position is instantiated with the constituent's semantic interpretation. Since the same constructor function is called with different arguments a number of times in order to construct an analysis incrementally, an argument is included in every constructor function that allows a "result so far" to be passed in and augmented. Its default value, which is used the first time the constructor function is executed, is the representation associated with the corresponding type in the absence of any arguments being instantiated. Each time the constructor function is executed, each of its arguments that are instantiated are first checked to be certain that the structures they are instantiated with match all of the type restrictions on all of the slots that are bound to that argument. If they are, the instantiated arguments' structures are inserted into the corresponding slots in the "result so far". Otherwise the constructor function fails.

Interleaving syntactic and semantic analysis in this

way allows semantic selectional restrictions to eliminate some of the ambiguity generated by the broad coverage syntactic grammar. It also allows the parser to avoid wasting time parsing portions of an input text that it does not have sufficient semantic knowledge to interpret. It has this effect because the parser fails to analyze any constituent that it cannot construct both a syntactic analysis and a semantic analysis for. At any time, a subset of semantic types can be selected from the complete meaning representation in order to restrict the parser further based on context specific semantic expectations. Additional context-specific semantic restrictions can also be introduced to specific types on the fly for the same purpose. For example, the meaning representation allows the theme associated with the `*draw` frame to be any object, but if in a particular context it was only important for the parser to interpret statements about drawing vectors, a further restriction could be imposed on the corresponding slot.

Conclusions and Current Directions

This paper describes a design for and current progress towards the development of the DAIENU authoring tool set for facilitating the rapid development of robust language understanding interfaces.

The DAIENU tool set is part of the larger Knowledge Construction Dialogue Authoring Tool Suite developed in the context of the Atlas project to automate the authoring of all domain specific knowledge sources required to implement domain specific dialogue capabilities in the general Atlas architecture. A simplified prototype version of DAIENU has been already been built. The simplified version does not yet include the interactive refinement stage described above. Furthermore, it does not yet make use of the full power offered by the CARMEL core understanding component. As mentioned, the prototype version of DAIENU was recently used to develop a language understanding interface sufficient for allowing students to navigate through lines of reasoning for 50 physics rules targeting all areas of Newtonian mechanics in the Atlas-Andes Physics tutoring system. The resulting language understanding interface required only 1 working day to complete using the prototype version of DAIENU. It consists of 714 basic patterns organized into 148 abstract conceptual classes. The corresponding planning interface required 2 man months of authoring time and resulting in 1815 APE operators. The resulting authored version of Atlas-Andes is currently being pilot tested with naive human subjects.

Acknowledgements

This research was supported by NSF grant number 9720359 to CIRCLE, the Center for Interdisciplinary Research on Constructive Learning Environments at the University of Pittsburgh and Carnegie-Mellon University.

The author would like to acknowledge Pam Jordan for her cooperation in the development of some aspects of the current prototype version of the KCD Authoring Tool Suite and Alon Lavie for his involvement in the early development of the LCFLEX robust parser and in the writing of documents describing earlier incarnations of the parsing work reported here.

This paper is dedicated to Eric and Rachel Rosé, who are most dear to the author's heart, and without whose encouragement, inspiration, and support the author would have certainly lost hope long ago.

References

Bresnan, J. 1982. *The Mental Representation of Grammatical Relations.* The MIT Press.

Bruninghaus, S., and Ashley, K. D. 1999. Toward adding knowledge to learning algorithms for indexing legal cases. In *Proceedings of the 17th International Conference on Artificial Intelligence and Law.*

Burgess, C., and Lund, K. 1997. Modeling parsing constraints with high-dimensional context space. *Language and Cognitive Processes* 12(2):177–210.

Burgess, C.; Livesay, K.; and Lund, K. 1998. Explorations in context space: Words, sentences, discourse. *Discourse Processes* 25(2):211–257.

Chi, M. T. H.; Bassok, M.; Lewis, M. W.; Reimann, P.; and Glaser, R. 1989. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science* 13:145–182.

Chi, M. T. H.; de Leeuw, N.; Chiu, M.; and Lavancher, C. 1994. Eliciting self-explanations improves understanding. *Cognitive Science* 18:439–477.

Cruse, D. A. 1986. *Lexical Semantics.* Cambridge University Press.

Dalrymple, M.; Lamping, J.; and Saraswat, V. 1993. LFG semantics via constrants. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL-93).*

Dalrymple, M. 1999. *Semantics and Syntax in Lexical Functional Grammar.* The MIT Press.

Elmi, M., and Evens, M. 1998. Spelling correction using context. In *Proceedings of COLING/ACL 98.*

Freedman, R.; Rosé, C. P.; Ringenberg, M. A.; and VanLehn, K. 2000. Its tools for natural language dialogue: A domain-independent parser and planner. In *Proceedings of the Intelligent Tutoring Systems Conference.*

Freedman, R. K. 2000. Plan-based dialogue management in a physics tutor. In *Proceedings of the 6th Applied Natural Language Processing Conference.*

Glass, M. S. 1999. *Broadening Input Understanding in an Intelligent Tutoring System.* Ph.D. Dissertation, Illinois Institute of Technology.

Glickman, O., and Jones, R. 1999. Examining machine learning for adaptable end-to-end information extraction systems. In *Proceedings of the AAAI Workshop on Machine Learning for Information Extraction.*

Grishman, R.; Macleod, C.; and Meyers, A. 1994. COMLEX syntax: Building a computational lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94).*

Halliday, M. A. K. 1985. *An Introduction to Functional Grammar.* Edward Arnold: A division of Hodder and Stoughton.

Halvorsen, P. K. 1997. Situation semantics and semantic interpretation in constraint-based grammars. Technical Report 101, CSLI: Stanford University.

Helmbold, D. P., and Schapire, R. E. 1997. Predicting nearly as well as best pruning decision tree. *Machine Learning* 27(1).

Kakes, D., and Morris, T. 1996. Efficient incremental induction of decision trees. *Machine Learning* 24(3).

Laham, D. 1997. Latent semantic analysis approaches to categorization. In *Proceedings of the Cognitive Science Society.*

Landauer, T. K.; Foltz, P. W.; and Laham, D. 1998. Introduction to latent semantic analysis. To Appear in *Discourse Processes.*

Lavie, A., and Rosé, C. P. 2000. Optimal ambiguity packing in unification-augmented context-free grammars. In *Proceedings of the International Workshop on Parsing Technologies.*

Martin, J. K. 1997. An exact probability metric for decision tree splitting. *Machine Learning* 28(2/3).

Quinlan, J. R. 1990. Probabalistic decision trees. In Kodratoff, Y., and Michalski, R., eds., *Machine Learning: An Artifical Intelligence Approach.* Morgan Kaufman.

Riloff, E., and Shephard, J. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing.*

Riloff, E. 1996. Using learned extraction patterns for text classification. In Wermter, S.; Riloff, R.; and Scheler, G., eds., *Connectionist, Statistical, and Symbolic Approaches for Natural Language Processing.* Springer-Verlag.

Rosé, C. P., and Lavie, A. to appear. Balancing robustness and efficiency in unification augmented context-free parsers for large practical applications. In Junqua, J. C., and Noord, G. V., eds., *Robustness in Language and Speech Technologies.* Kluwer Academic Press.

Rosé, C. P., and Levin, L. S. 1998. An interactive domain independent approach to robust dialogue interpretation. In *Proceedings of COLING/ACL 98.*

Rosé, C. P., and Waibel, A. 1997. Recovering from parser failures: A hybrid statistical/symbolic approach. In Klavans, J., and Resnik, P., eds., *The Balancing Act: Combining Symbolic and Statistical Approaches to Language Processing.* The MITPress.

Rosé, C. P.; Moore, J. D.; VanLehn, K.; and Allbritton, D. 2000. A comparative evaluation of socratic versus didactic tutoring. Technical Report LRDC-BEE-1, Learning Research and Development Center, University of Pittsburgh.

Rosé, C. P.; Di Eugenio, B.; and Moore, J. D. 1999. A dialogue based tutoring system for basic electricity and electronics. In *Proceedings of Artificial Intelligence in Education.*

Rosé, C. P. 1997. *Robust Interactive Dialogue Interpretation.* Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University.

Rosé, C. P. 2000a. A framework for robust semantic interpretation. In *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics.*

Rosé, C. P. 2000b. A syntactic framework for semantic interpretation. In *Working Notes for the ESSLLI Workshop on Linguistic Theory and Grammar Implementation.*

Tomita, M. 1987. An Efficient Augmented Context-free Parsing Algorithm. *Computational Linguistics* 13(1-2):31–46.

Tomita, M. 1990. The Generalized LR Parser/Compiler - Version 8.4. In *Proceedings of International Conference on Computational Linguistics (COLING'90)*, 59–63.

Utgoff, P. E.; Berkman, N. C.; and Clouse, J. A. 1997. Decision tree induction based on tree restructuring. *Machine Learning* 29(1).

van den Berg, M.; Bod, R.; and Scha, R. 1994. A corpus based approach to semantic interpretation. In *Proceedings of the Ninth Amsterdam Colloquium.*

van Genabith, J., and Crouch, D. 1996. Direct and underspecified interpretations of lfg f-structures. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96).*

VanLehn, K.; Freedman, R.; Jordan, P.; Murray, C.; Osan, R.; Ringenburg, M.; Rosé, C. P.; Schultze, K.; Shelby, R.; Treacy, D.; Weinstein, A.; and Wintersgill, M. 2000. Fading and deepening: The next steps for andes and other model-tracing tutors. In *Proceedings of the Intelligent Tutoring Systems Conference.*

VanLehn, K.; Jones, R. M.; and Chi, M. T. H. 1992. A model of the self-explanation effect. *The Journal of the Learning Sciences* 2(10):1–59.

VanLehn, K.; Siler, S.; and Baggett, W. 1998. What makes a tutorial event effective. In *Proceedings of the Twenty-first Annual Conference of the Cognitive Society.*

Wiemer-Hastings, P.; Graesser, A.; Harter, D.; and the Tutoring Research Group. 1998. The foundations and architecture of autotutor. In Goettl, B.; Halff, H.; Redfield, C.; and Shute, V., eds., *Intelligent Tutoring Systems: 4th International Conference (ITS '98).* Springer Verlag. 334–343.