Logic-Based Natural Language Understanding in Intelligent Tutoring Systems

PhD Thesis

Octav Popescu

Language Technologies Institute
School of Computer Science
Carnegie Mellon University

Thesis Committee:
Kenneth Koedinger, Chair
Jaime Carbonell
Lori Levin
Kurt VanLehn, University of Pittsburgh
Michael Kohlhase, International University Bremen

Abstract

High-precision Natural Language Understanding is needed in Geometry Tutoring to accurately determine the semantic content of students' explanations. The thesis presents a Natural Language Understanding system developed in the context of the Geometry Cognitive Tutor. The system combines unification-based syntactic processing with Description Logic based semantics to achieve the necessary accuracy level.

The thesis examines in detail the main problem faced by a natural language understanding system in the geometry tutor, that of accurately determining the semantic content of students' input. It then reviews alternative approaches used in Intelligent Tutoring Systems, and presents some difficulties these approaches have in addressing the main problem.

The thesis proceeds to describe the system architecture of our approach, as well as the compositional process of building the syntactic structure and the semantic interpretation of students' explanations. The syntactic and semantic processing of natural language are described in detail, as well as solutions for specific natural language understanding problems, like metonymy resolution and reference resolution.

The thesis also discusses a number of problems occurring in determining semantic equivalence of natural language input and shows how our approach deals with them. The classification performance of the adopted solution is evaluated on data collected during a recent classroom study and is compared to a Naïve Bayes approach.

The generality of our solution is demonstrated in a practical experiment of porting the approach to a new semantic domain, Algebra. The thesis discusses the changes needed in the new implementation, the time effort required, and presents the classification performance in the new domain.

Finally, the thesis provides a high level Description Logic view of the presented approach to semantic representation and inference, and talks about the possibility to implement it in other logic systems.

Table of Contents

Abstract *3*

Table of Contents *5*

Chapter 1 Utility of Natural Language Understanding in Cognitive Tutoring *9*

Chapter 2 The Approach and Contributions *17*

Chapter 3 Natural Language Understanding in Geometry Tutoring - Determining Semantic Content *19*

3.1. Equivalence of semantic content over various ways of expression **20**
3.1.1. Variation of syntactic structure 20
3.1.1.1. Passive versus active 20
3.1.1.2. Adjunct phrase attachment 21
3.1.1.3. Clauses and other cases 22
3.1.2. Variation of content words 23
3.1.2.1. Synonyms 23
3.1.2.2. Use of generic concepts 23
3.1.2.3. Generic relations 24
3.1.2.4. Use of definitions instead of specific concepts or relations 25
3.1.2.5. Ellipses 25
3.1.2.6. Anaphora 26
3.1.3. Combinations 26

3.2. Other problems that require semantic solutions **27**
3.2.1. Plurals: Distributive vs. collective 27
3.2.2. Syntactic ambiguity: Prepositional phrase attachment 28
3.2.3. More structural ambiguity: Noun-noun compounds 29
3.2.4. Apparent semantic ill-formedness: Semantic structure does not follow syntactic structure: metonymy, sets 29
3.2.5. Reference resolution 30

Chapter 4 Alternative Approaches *33*

4.1. Statistical based approach - Autotutor **33**
4.1.1. Limitations of statistical approaches 34

4.2. Semantic grammars **35**
4.2.1. Sophie 35
4.2.2. Nespole! 35

4.2.3. Limitations of the semantic grammars approach 36
4.3. Finite state syntax approach - CIRCSIM-Tutor 36
4.3.1. Limitations of the finite state syntax approach 37
4.4. Deep syntactic/frame-based semantic approach 37
4.4.1. Atlas-Andes tutor 37
4.4.2. Limitations of Atlas-Andes system 38
4.4.3. KANT/KANTOO machine translation system 39
4.4.4. Challenges for the KANT system 39
4.5. Deep-syntactic/logic-based semantic approach – Gemini 39
4.5.1. Limitations of the Gemini system 40
Chapter 5 System Description 41
5.1. System architecture 41
5.2. Syntactic processing 43
5.2.1. Chart parser 43
5.2.2. Active chart 43
5.2.3. Unification grammar 45
5.2.4. Lexicon 46
5.2.5. Feature structure unifier 46
5.3. Semantic processing 48
5.3.1. Description Logic 48
5.3.2. Upper Model and Geometry knowledge bases 48
5.3.3. Semantic representation 51
5.3.4. Example of semantic structure 52
5.3.5. Linguistic inference 54
5.3.6. Semantic contexts 57
5.3.7. Example of compositional build 58
5.3.8. Reference resolution 60
5.3.9. Metonymy resolution 63
5.3.10. Classification 63
Chapter 6 Advantages of Logic-Based Approach over Alternatives 67
6.1. Natural, consistent modeling 67
6.2. Advantages in determining the semantic content of sentences 68
6.2.1. Uniqueness of semantic content over various ways of expression 68
6.2.1.1. Variation of syntactic structure 68
6.2.1.2. Variation of content words 73
6.2.2. Structural ambiguity 76
6.2.3. Reference resolution disambiguation 76
Chapter 7 Evaluation of NLU performance 79
7.1. Evaluation method 79
7.2. Evaluation results 81

7.3. Robustness issues 84
7.3.1. Grammatical problems 86
7.3.1.1. Agreement (23 cases) 86
7.3.1.2. Extra words (10 cases) 86
7.3.1.3. Missing words (27 cases) 86
7.3.1.4. Wrong words (38 cases) 87
7.3.1.5. Wrong references (5 cases) 87
7.3.1.6. Robustness results on grammatical problems 87
7.3.2. Semantic problems 89
7.3.2.1. Number problems (17 cases) 89
7.3.2.2. Proper semantic problems (40 cases) 89
7.3.2.3. Robustness results on semantic problems 90
7.4. Comparison of performance with a Naïve Bayes approach 91
7.5. Comparison of performance with K-Nearest Neighbor approach 93
Chapter 8 Applying Logic-Based NLU to the Algebra Domain 95
8.1. Explanation in equation solving 95
8.1.1. Evaluation of development effort 95
8.1.2. Evaluation of data needs 96
8.2. Modifications required by the new application 96
8.2.1. Lexicon additions 96
8.2.2. Knowledge base additions 97
8.2.3. Metonymy resolution 98
8.2.4. Classification taxonomy 100
8.2.5. Grammar modifications 101
8.2.6. Summary of modifications 103
8.3. Evaluation of the classification accuracy of the new application 103
8.4. Evaluation of time effort 105
8.5. Conclusions of the development of the Algebra application 107
Chapter 9 Implementing the System in Description Logic 109
9.1. Description Logic features needed for knowledge base modeling 109
9.1.1. Concepts 109
9.1.2. Concept negations 110
9.1.3. Concept conjunctions 111
9.1.4. Concept disjunctions 111
9.1.5. Concept equalities 112
9.1.6. Concept inclusions 114
9.1.7. Trigger rules 114
9.1.8. Roles 116
9.1.9. Universal role restrictions/quantifications 116
9.1.10. Full existential role restrictions/quantifications 116
9.1.11. Functional role restrictions 117
9.1.12. Unqualified number restrictions 118

9.1.13. Role hierarchies/inclusions 119
9.1.14. Role definitions/equalities 119
9.1.15. General role axioms 120
9.1.16. Role conjunctions 120
9.1.17. Role disjunctions 121
9.1.18. Role complements 121
9.1.19. Role transitivity 121
9.1.20. Role compositions 121
9.1.21. Inverse roles 122
9.1.22. Domain and range role restrictions 123
9.1.23. Role-value maps (agreements) 123
9.1.24. ':relates' constructs 124
9.1.25. Concrete domains 125

9.2. Representing semantic structures in Description Logic 126
9.2.1. Linguistic inference in Description Logic 127
9.2.2. Reference resolution 128
9.2.3. Semantic contexts in Description Logic 129

9.3. Inference services used in the NLU system 129
9.3.1. Concept subsumption/classification 130
9.3.2. Concept satisfiability and equivalence 130
9.3.3. Concept disjointness 130
9.3.4. Individual consistency 131
9.3.5. Individual realization/classification 131
9.3.6. Role specialization 132
9.3.7. Constraint propagation on individuals 132

9.4. Portability to other Logic Systems 133
9.4.1. PowerLoom 133
9.4.1.1. Logic language 133
9.4.1.2. Semantic representation and contexts 137
9.4.1.3. Reasoning services 137
9.4.2. Racer 138
9.4.2.1. Logic language 138
9.4.2.2. Semantic representation and contexts 142
9.4.2.3. Reasoning services 142

Chapter 10 Conclusions and Future Work 145

10.1. Semantic repair mechanism 145
10.1.1. Implicit references 145
10.1.2. Malformed sentences 146
10.1.3. Incompleteness in knowledge base coverage 146
10.1.4. Semantic vicinity conceptual search 147

Bibliography 149

Chapter 1 Utility of Natural Language Understanding in Cognitive Tutoring

The Pittsburgh Advanced Cognitive Tutor Group (PACT) at the Human-Computer Interaction Institute of Carnegie Mellon University researches the use of cognitive tutors, a kind of Intelligent Tutoring Systems, for middle school and high school education. So far the group has developed a number of cognitive tutors for Algebra and Geometry. These tutors are currently used in over 1000 high and middle schools around the country to help students master these subjects.

In previous evaluation studies, Koedinger & al (1997) have shown that the cognitive tutors are successful in raising high school students' test scores in Algebra. However, other studies (Bloom, 1994) have shown that there is still a considerable gap between the effectiveness of current cognitive tutor programs and human tutors.

One main difference between cognitive tutor systems and human tutors is that most current cognitive tutors only teach students how to solve particular problems in their respective field. That is, even if the ultimate goal is to teach students basic generic principles of the domain of focus, this is somewhat hidden behind the scenes. What the tutors actually do is to propose problems to students, and then check students' solutions. They also provide context-sensitive hints at each step in solving the problem, trying to direct the students towards the correct solution.

However, generally they do not ask students to explain or justify their answers. Like asking for instance, "why did you do this step?" or "what rule can you apply next and why?" or "what does this rule mean?" In the few cases where tutors do ask students for explanations, they do it either by having the students choose an explanation out of a given list, like in the current version of the Geometry Cognitive Tutor (Aleven & al, 1999), or by making them build up an answer by choosing elements of a fixed template, like in Ms. Lindquist (Heffernan & Koedinger, 2000).

On the other side, human tutors seem to excel in engaging students in dialogs aimed to make students think about the reasons behind the solution steps. This reasoning process has the potential to improve students' understanding of the domain, resulting in knowledge that generalizes better to new problems (Chi & al, 2001). This difference might also be the main explanation beneath the gap mentioned above. Under this hypothesis, the main goal in developing the next generation of intelligent cognitive tutors is to make them be able to carry on tutoring dialogs with students at a deeper explanation level.

Rather than being a flaw with the design of current cognitive tutors, the lack of a dialog at the explanation level is the result in part of a lack of good technology for natural language processing. A system component that would understand students' input expressed in natural language is one of the key components in implementing such a tutorial dialog around student explanations. Some current intelligent tutoring systems like Autotutor (Wiemer-Hastings & al, 1999), Circsim-Tutor (Glass, 1997), and Atlas/Andes (Jordan & al, 2001) do have some natural language processing capabilities. However, these systems rely on either statistical processing of language, identifying keywords in language, or shallow syntactic processing.

This thesis argues that none of these approaches is good enough for use in a highly formalized domain such as mathematics, because they do not go deep enough into processing students' explanations. In order to understand an explanation in the domain of mathematics in full, a natural language understanding system needs to perform a considerable amount of inference based on knowledge of the domain. This inference is desirable mainly because of the high degree of precision needed in determining the semantic content of students' utterances. There are many different ways to express the same semantic content, which have to be recognized as being equivalent. On the other side, there are many cases when a small distinction makes the difference between a precisely stated correct explanation and a nearly correct explanation, and such cases need to be detected, in order to be corrected.

This inference process has to be done in a consistent way, so no unwarranted conclusions are derived from the text, which would alter the tutoring process. And the inference has also to be done robustly, in an environment of imprecise or ungrammatical language as is uttered more often than not by high school students. Our hypothesis is that such a system needs to be based on a logic system in order to be able to achieve the level of understanding and inference required. In this thesis we describe how to build such a system, and we evaluate its performance.

We are building this understanding system in the context of the Geometry Cognitive Tutor. In line with the general approach mentioned above, the Geometry Cognitive Tutor assists students in learning by doing as they work on geometry problems on the computer. As a kind of Cognitive Tutor (Anderson & al, 1995), this system is based on an underlying cognitive model, implemented as an ACT-R production system (Anderson & Lebiere, 1998), of both novice and ideal student knowledge. This model is used to monitor student performance and to provide assistance just when students need it and in a context that demands it. The tutor maintains a detailed assessment of each student's skills, which is used to select appropriate problems and determine pacing. Currently the Geometry Cognitive Tutor is in regular use (two days per week) in about 150 schools around the US. The tutor curriculum consists of six extensive units: Area, Pythagorean Theorem, Angles, Similar Triangles, Quadrilaterals, and Circles.

Figure 1-1. Real case dialog – step 4 start

The development process was focused on the Angles unit. In this unit, students analyze geometric figures with given properties (e.g., parallel lines or isosceles triangles) and solve problems by inferring measures of angles and segments. Currently each step in a problem involves both entering a measure value (e.g., 50 degrees) and the name of a "reason", a geometric definition or theorem (e.g., Triangle Sum) that explains why the entered value must be what it is. Thus, the current tutor tries to bypass the need for natural language understanding by requiring students to just enter a reason name either by typing or by selecting it from an on-line glossary. It has been shown (Aleven & al, 1999) that this process improves students' understanding and scores in subsequent tests, relative to a tutor where students do not provide any kind of explanations. However, our hypothesis is that full natural language understanding (NLU) capabilities will enhance student performance even further.

To explore this hypothesis we have built an NLU version of the Geometry Cognitive Tutor, called the Geometry Explanation Tutor. This tutor works similarly to the previous one, by presenting students with geometry problems and asking them to solve these problems. At each step in the solving process the students are required to provide an answer (usually the measure of an angle they have to determine) and a reason for their answer. Unlike in the previous tutor where students were providing this reason by selecting it from a glossary, in the Geometry Explanation Tutor the reason has to be a natural language expression of the corresponding definition or theorem.

As an example of student-tutor interaction let's take a *real dialog* taken from a pilot study conducted in the Spring of 2003. During the study, the Geometry Explanation Tutor was used for about a week in a suburban junior high school in the Pittsburgh area, as part of a 9th-grade Integrated Mathematics II course.

At the fourth step in solving the problem in Figure 1-1 the student has computed the value of angle ORC to be 77.75 degrees and moved on to provide a reason for it. Thinking the same reason used in previous steps applies again, at the first try the student inputs:

(1) `in an isosceles triangle, base angles are the same`

This is a correct sentence for the Isosceles Triangle theorem. However this is not the theorem that applies in this particular case. Accordingly the tutor colors the explanation in red and pops a hint window with the message:

(2) `You gave a correct statement of a geometry rule, but that rule is not needed here. Please focus on a different rule.`

The student makes a second try, typing:

(3) `interior angles are congruent`

This is a partial expression of possibly two different theorems, the Isosceles Triangle theorem and the Alternate Interior Angles theorem. However, none of these applies to the current case, so the tutor replies again with a message directing the student to try something else:

(4) `You may be thinking of the wrong geometry rule. Please focus on a different rule.`

Figure 1-2. Real case dialog – step 4 middle

The student keeps thinking that interior angles are involved, but at the next step provides contextual information that characterizes the situation more precisely:

```
(5)   when two parallel lines are cut by a transversal, the
      interior angles are congruent
```

Recognizing that this time the student is closer to a good explanation, but he still focuses on the wrong type of angles, the tutor gives an encouraging reply, and at the same time points to the mistake (see Figure 1-2):

```
(6)   Your explanation looks good. However, you do not need to
      talk about interior angles here. So it is better to delete
      or change this part.
```

The student takes the hint, and trying to find the correct types of angles, switches '`interior`' to '`exterior`':

```
(7)   when two parallel lines are cut by a transversal, the
      exterior angles are congruent
```

However, this is still not the correct kind of angles involved in solving the step, so the tutor replies with a hint similar to the previous one:

```
(8)   Yes, you are on the right track. One thing, though:
      mentioning exterior angles here is not correct. So you
      should probably change that part of your explanation.
```

Finally, the student figures out that the angles involved in this step are corresponding angles, so he corrects his previous attempt with:

```
(9)   when two parallel lines are cut by a transversal, the
      corresponding angles are congruent
```

The tutor accepts the explanation by coloring it in black and allowing the student to move to the next step, as seen in Figure 1-3.

Figure 1-3. Real case dialog – step 4 end

The entire dialog, together with the classification of the student's input provided by the tutor and by two human raters are given in Table 1-1 below.

Part.	Explanation/Feedback	System's Class	Humans' Class
Student	in an isosceles triangle, base angles are the same	(CONGRUENT-BASE-ANGLES-OF-ISOSCELES-TRIANGLE)	(CONGRUENT-BASE-ANGLES-OF-ISOSCELES-TRIANGLE)
Tutor	You gave a correct statement of a geometry rule, but that rule is not needed here. Please focus on a different rule.		
Student	interior angles are congruent	(CONGRUENT-INTERIOR-ANGLES)	(CONGRUENT-INTERIOR-ANGLES)
Tutor	You may be thinking of the wrong geometry rule. Please focus on a different rule.		
Student	when two parallel lines are cut by a transversal, the interior angles are congruent	(PARALLEL-LINES-INTERSECTED-BY-TRANSVERSAL CONGRUENT-INTERIOR-ANGLES)	(PARALLEL-LINES-INTERSECTED-BY-TRANSVERSAL CONGRUENT-INTERIOR-ANGLES)
Tutor	Your explanation looks good. However, you do not need to talk about interior angles here. So it is better to delete or change this part.		
Student	when two parallel lines are cut by a transversal, the exterior angles are congruent	(PARALLEL-LINES-INTERSECTED-BY-TRANSVERSAL CONGRUENT-EXTERIOR-ANGLES)	(PARALLEL-LINES-INTERSECTED-BY-TRANSVERSAL CONGRUENT-EXTERIOR-ANGLES)
Tutor	Yes, you are on the right track. One thing, though: mentioning exterior angles here is not correct. So you should probably change that part of your explanation.		
Student	when two parallel lines are cut by a transversal, the corresponding angles are congruent	(PARALLEL-LINES-CONGRUENT-CORRESPONDING-ANGLES)	(PARALLEL-LINES-CONGRUENT-CORRESPONDING-ANGLES)
Tutor	[accepts as correct and complete explanation by means of implicit feedback]		

Table 1-1. Example of a successful student-tutor dialog from the pilot study

Chapter 2 The Approach and Contributions

In this thesis we present a natural understanding system that relies on a combination of technologies to achieve the necessary level of precision in understanding. The main parsing process is directed by a left-corner chart parser (Rosé & Lavie, 1999) that uses a full syntactic-driven LFG-oriented (Bresnan, 2001) unification-based grammar to model English. However, instead of building the semantics in an ad-hoc manner within the same unification-based formalism, the system creates a semantic representation of the meaning in a Description Logic system called Loom (MacGregor, 1991). Loom is designed to facilitate the development of reasoning systems based on a subset of first-order logic. The semantic representation makes use of contextual knowledge provided by a model of geometry developed using the conceptual language of Loom. The model ensures the logical consistency of the semantic representations and facilitates the reasoning processes needed for attaining the necessary degree of understanding.

We chose this approach because it has the potential to be the most adequate considering the requirements of the natural language understanding task in the context of the Geometry Tutor. There are two main aspects that make this task more difficult than in previous tutor systems. First, we do not analyze answers to tutor's questions, but free form explanations. Second, this analysis is performed within a domain with a highly formalized model like mathematics. These aspects are further strengthened by the generic requirement of high precision common to all tutoring tasks.

The main difference between answers and explanations relies in the complexity of the natural language input that needs to be analyzed. In the case of answers, the input is usually composed of only a couple of semantic elements, most often combined together by an obvious logical relation like conjunction (Glass, 2001). In case of free form explanations, an input sentence can usually be composed of between 5 and 15 semantic elements connected through various relations specific to the domain of discourse. Of course questions also can be designed to require highly elaborated answers. However previous tutors (Glass, 2000) made a design principle from avoiding them, in a quest to avoid the complexity of full natural language understanding.

A second difference between questions and explanations relies in the fact that in case of questions, the tutor already knows what the right answers are. Then it can use this information in trying to determine whether to accept the input as an acceptable answer or not, even in absence of a thorough understanding of the input. There are also open questions that could take an unrestricted number of acceptable answers. But again existing tutors have systematically avoided such types of questions. Explanations are like answers to a generic open question "Why?" Thus, even if the tutor knows what is the

right explanation for each specific case, it cannot use this in the analysis process. The task here is not just to determine whether to accept the explanation as good or not, but to determine the actual content of the explanation, in order to be able to take appropriate corrective action.

The context of a mathematical domain of discourse like geometry requires in particular a high degree of natural language understanding, in order to be able to assess whether the language input captures the precise mathematical concepts and relations required to express theorems and definitions of the domain. In particular, this task has to be performed while dealing with the ambiguity and impreciseness of common language. We argue that natural language understanding of this kind is hard to be performed with methods that lack inference capabilities. And in order to perform such inferences reliably the tutor needs to rely on extensive modeling of the knowledge in the domain of discourse.

One way to look at the NLU process is as a classification task, where the goal is to get one or more classes that accurately describe the semantic content of the sentence. Viewed this way, the domain at hand is characterized by a large number of classes, of the order of hundreds, needed to cover all meaningful differences in expressing the Geometry theorems and principles. At the same time these classes are often very close semantically, so the classifier has to be able to make fine distinctions among such cases.

The main contribution of this work is the development of a working natural language understanding system that demonstrates how to systematically built a semantic representation for natural language sentences, in a compositional way, within the framework of a Description Logic system. This system is capable to achieve the degree of precision in understanding required by the task at hand, and it does so in the real world application of high school geometry tutoring. Although the system does show a number of robustness features, the main focus of the work has been on accuracy.

The system provides an effective and theoretically sound way to incorporate a Description Logic system in the NLU process. And at the same time if provides a highly reusable architecture that facilitates its portability to new domains. In the process of developing the system, a number of difficult problems of natural language understanding were examined and working solutions to those problems were developed, solutions that are detailed in the dissertation. An evaluation of the portability of the chosen approach to new domains of discourse, through an actual implementation in the domain of Algebra is also present.

The system builds Description Logic-based semantic representations for natural language sentences, representations that can be manipulated in various ways to extract the meaningful information from the text. One such way is to use it as basis for classification. The thesis also provides an evaluation of the classification performance of system based on actual classroom data,

Finally the dissertation provides a high-level detailed analysis of the features and inferences services needed to implement our approach in a generic description logic system, and discusses the possibility of porting it to other state of the art systems.

Chapter 3 Natural Language Understanding in Geometry Tutoring - Determining Semantic Content

In many domains that process natural language input, like for instance in processing information requests, the task at hand can be performed with a reasonable degree of success based only on a relatively shallow degree of understanding, like that provided by statistical processing of input, or syntactic parsing. However, in tutoring domains, especially on topics like mathematics, such an approach is not likely to yield good results. In such a domain it is important to not only assess whether students appear to have the right idea, but also whether they can articulate this idea in full. In other words, it is important to determine if the student understands the crucial components of the idea (theorem or definition) that delineate when and how it applies and when it does not. We argue that such a determination needs a deeper level of understanding. Some of the linguistic problems that make this process difficult, and which we have dealt with within the work on this thesis, are described below.

In lack of an extensive modeling of the domain of discourse most systems rely on one or both of two kinds of information in order to determine the semantic content of a sentence: syntactic structure and choice of words. Many times syntactic structure and/or choice of words alone do not reflect precisely the actual semantic content of the sentence. The mapping from surface language to meaning is multi-multi. On one hand there is usually a considerable variety of different ways to express the same meaning using different words and different syntactic constructions. On the other hand the same sequence of words can have different meanings in different contexts. The problem is further compounded by a number of linguistic phenomena that a system needs to deal with, like ambiguity, ungrammaticalities, anaphora resolution, and metonymy.

Some of the specific problems that make the mapping process difficult for alternative approaches to NLU are detailed below, with examples from the geometry-tutoring domain. Unlike in the rest of thesis, some of the examples in this chapter are not taken from a real corpus, but are rather made up to better illustrate the point discussed. However all problems are present in the corpus used for evaluation of system performance in Chapter 7, and an indication is given on their frequency.

3.1. Equivalence of semantic content over various ways of expression

In order for a system to react reliably to the semantic content of natural language, it has to be able to determine accurately when various input word sequences are equivalent with respect to semantic content and when not.

Statistical systems usually try to deal with the problem through a process of statistical classification. Such a system has several major problems. First, the basic statistical language model may not be able to capture some of the variations in the input string that determine significant differences in the semantic content. Trying to refine the statistical model can lead to an increasing number of parameters that have to be learned. Which leads to another problem: the need for huge amounts of pre-classified training data in order to get good results. In many cases such data is not available, and/or requires a considerable effort to gather and classify.

Symbolic systems that rely solely on syntactic parsing of input language usually end up with different representations for the same meaning, when presented with highly different input strings. The problem is then that in absence of a mechanism that is able to use information about the domain of discourse, there is no reliable way to determine the semantic equivalence of these representations. Many times this equivalence relies on inference processes specific to the domain of discourse.

The determination of equivalence relations has to work robustly over variation of syntactic structure, variation of content words, or a combination of both.

3.1.1. Variation of syntactic structure

Even when the choice of content words is the same, the same meaning can be conveyed through a variety of syntactic structures. A few such cases are discussed below.

3.1.1.1. Passive versus active

One obvious case where syntactic structure does not lead to any significant difference in semantic content is that of passive versus active constructs. 150 sentences use passive constructs in our evaluation corpus of 700 sentences (21%).

(10) a) `Two intersecting lines form congruent vertical angles.`

b) `Congruent vertical angles are formed by two intersecting lines.`

There are two main ways to deal with passive constructs. One is to model the construct in the language model. The other is to completely ignore sentence structure and functional words, as "bag of words" statistical models do.

Symbolic models based on syntactic language grammars can easily deal with this particular case. This is because this particular variation is a general linguistic phenomenon, which can be captured in an elegant and general way in the grammar.

3.1.1.2. Adjunct phrase attachment

Another example where variation of syntactic structure may not lead to a change in meaning is that of prepositional phrase attachment.

(11) a) In a triangle angles opposite to congruent sides are congruent.

b) Angles in a triangle opposite to congruent sides are congruent.

c) Angles opposite to congruent sides in a triangle are congruent.

d) Angles opposite to congruent sides are congruent in a triangle.

All these cases are semantically equivalent, since all elements are actually '`in a triangle`'. We had 52 such cases in our 700 sentences corpus. This is not always the case, as it can be seen in the examples below:

(12) a) The measure of an angle formed by bisecting another angle is equal to half the measure of the bisected angle.

b) The measure of the bisected angle is equal to half the measure of an angle formed by bisecting another angle.

In this case only the first sentence states a valid Geometry theorem, even if the only structural difference consists in switching two prepositional phrases. Similarly, in the examples below the attachment of an adjective phrase is changed, resulting in two sentences that represent two reciprocal theorems.

(13) a) Angles opposite to congruent sides in a triangle are congruent.

b) Sides in a triangle opposite to congruent angles are congruent.

Statistical models that rely only on the "bag of words" in a sentence, and ignore word sequence (like latent semantic analysis, see section 4.1) can deal easily with sentences in example (11), since they just ignore the difference in structure. However, they have problems when this difference in structure is actually significant, like in examples (12), and (13).

Unlike the previous case, syntactic approaches alone are also unable to deal with such a situation. Here the determination of whether the variation of syntactic structure leads to a significant change in semantic content relies on specific knowledge from the domain of discourse. In such cases a semantic model of the domain of discourse is actually needed to reliably determine the semantic equivalence of the two examples.

Ambiguity problems concerning prepositional phrase attachment are discussed in section 3.2.2 below.

3.1.1.3. Clauses and other cases

The problem of semantic equivalence of various syntactic structures is made even more apparent if we consider more complex examples, where the syntactic structure is even more different, like when using constructs specific to the domain of discourse:

(14) a) The measures of these two angles are equal.

b) These two angles are equal in measure.

c) These two angles have equal measures.

d) These two angles measure the same.

Knowledge about the semantics of 'equal' and 'measure' is involved in determining that 'equal in measure' means the same thing as 'measures … are equal'. Similar knowledge is needed to figure out that 'are equal in measure' is equivalent to 'have equal measures'. Knowledge about the meaning of 'the same' as equivalent to 'equal' and the verb 'measure' as equivalent to 'has a measure' is needed to establish the equivalence of 'measure the same' to the previous sentence.

Syntactic parsers would have a difficult time trying to get the same output out of these four examples, since the equivalence relation relies heavily on the semantics of the words involved. One possible approach for such cases is that of semantic grammars (see section 4.2 for a more detailed discussion). In such an approach the equivalence would be specified explicitly for each such case, which leads to considerable development effort.

The use of relative and subordinate clauses can also lead to a large variety of syntactic structures without a significant change in meaning.

(15) a) The sum of the measures of a pair of complementary angles is 90 degrees.

b) If two angles are complementary, then the sum of their measures is 90 degrees.

c) The angle sum is 90, because they are complementary angles.

d) Complementary angles are angles whose measures sum to 90 degrees.

For example here are a few sentences that all express the same theorem about complementary angles using either a single clause sentence in a), a conditional clause in b), a subordinate clause in c), or a relative clause in d). Our 700 sentence corpus contains 140 relative clauses, 109 subordinate clauses, and 29 conditional clauses, for a total of 278 cases, or about 40% of the sentences.

Recognizing such equivalencies can be done with some degree of success by ignoring sentence structure all together. Being able to accurately distinguish between cases where different structure leads to significant difference in meaning and cases where it does not requires a combination of a syntactic approach with a domain-specific semantics model.

Another similar example is shown below. All these sentences taken from our corpus are realizations of the same theorem about base angles in an isosceles triangle. There are

some differences in the way the basic implication is expressed, but those can be ignored for tutoring purposes.

(16) a) The base angles of an isosceles triangle are congruent.

b) They are base angles, which are congruent in an isosceles triangle.

c) In an isosceles triangle the base angles are congruent.

d) This is an isosceles triangle, so its base angles are congruent.

e) Its base angles are congruent because it's an isosceles triangle.

f) They are congruent angles in an isosceles triangle and they are the base angles.

3.1.2. Variation of content words

3.1.2.1. Synonyms

Many times differences in the content words used in the sentence do not make any difference at the meaning level. A first such case is that of synonyms.

(17) a) The angles in a triangle add to 180 degrees.

b) The angles in a triangle add up to 180 degrees.

c) The angles in a triangle sum up to 180 degrees.

If the words used as synonyms are synonyms in all contexts, the problem is easily solvable in all symbolic approaches, by just defining the words as having the same meaning.

There are cases however when different words are synonyms only in certain contexts. For instance:

(18) a) Angles ABC and BAC are equal.

b) Angles ABC and BAC are congruent.

Versus:

(19) a) The measures of angles ABC and BAC are equal.

b) *The measures of angles ABC and BAC are congruent.

Here the synonymy holds only when the objects involved in the relation are geometry objects, and it is not allowed when they are measures. Making this distinction accurately is hard in absence of a model of the semantics of the domain, Geometry in this case.

3.1.2.2. Use of generic concepts

One way to treat the previous example is to consider '`congruent`' as a more specific concept of '`equal`', which is constrained to apply only to geometry objects. This example is just an instance of a more general class of phenomena, when a generic term is

used instead of a more specialized term, in situations where the context makes clear the more specific term is actually meant. Another such example is:

(20) a) The line between points A and B measures 10 cm.

b) The segment between points A and B measures 10 cm.

In a strict geometrical sense sentence a) above could be considered incorrect, since by definition lines are infinite, and thus cannot have a measure. However in usual language '`line`' is used as a generic term denoting all kind of linear objects, including rays and segments. Then in sentence a) coherence constraints coming from the domain of discourse make clear that the only way the sentence can have a valid meaning is if the word '`line`' is used to name a segment. Under this interpretation sentence a) is equivalent semantically to sentence b). Any language model that lacks domain knowledge will have difficulties with such cases. In order to infer this equivalence, a system needs to be able to perform inferences specific to the domain of discourse at the right level of generality. At the same time the system needs to allow for easy maintenance of these inferences in the development process, while maintaining consistency of the model in the presence of hundreds of such constraints.

3.1.2.3. Generic relations

An even more general phenomenon closely related to the previous one is that of using very generic functional words in usual language to denote very precise relations among the concepts of the domain.

(21) a) The angles of a triangle sum to 180.

b) The angles in a triangle sum to 180.

c) The angles that are vertices of a triangle sum to 180.

(22) a) The angles of a linear pair sum to 180.

b) The angles that form a linear pair sum to 180.

c) The angles that are elements of a linear pair sum to 180.

In example (21) the proper explicit relation between the angles and the triangle is that the angles are the triangle's vertices. Many times such a relation is not explicitly specified in language, but rather implicitly conveyed through the use of a generic preposition, like '`of`' or '`in`'. Similarly, in example (22) the angles are actually the elements of the linear pair. However the relation is expressed either through a preposition, or through a generic verb like '`form`'.

Recovering the explicit relation and thus being able to determine that the three examples in each pair above are semantically equivalent requires once again a detailed model of the domain of discourse. Such a model could be trained into a statistical system or written into a semantic grammar, but only a logic-based model would be able to achieve the desired degree of generality.

3.1.2.4. Use of definitions instead of specific concepts or relations

Another situation that can be seen as a case of the above is when people use an explicit definition of a concept expressed in terms of more generic concepts, instead of the name of the more specific concept.

(23) a) In a triangle with two congruent sides the base angles are congruent.

b) In an isosceles triangle the base angles are congruent.

(24) a) Adjacent angles on a line sum to 180 degrees.

b) Linear angles sum to 180 degrees.

(25) a) Opposite angles that are formed by two intersecting lines are congruent.

b) Vertical angles are congruent.

The ability to recognize such examples as being semantically equivalent with the right degree of generality is conditioned by the possibility to model the definitions of those specific concepts within the framework of the system. Logic based approaches have a much better chance of capturing the right relationships correctly and generalizing to new cases.

3.1.2.5. Ellipses

Elliptical sentences provide another example where different sentence content leads to equivalent meaning. Most times in usual language, part of the content is not realized in the word string, but is left unspecified. In a number of cases part of this missing content can be recovered based on knowledge about the concepts involved.

(26) a) The angles are 90.

b) The angles are 90 degrees.

c) The measures of the angles are 90 degrees.

d) The measures of the angles are equal to 90 degrees.

Here both the measure units and the specification of the property of these angles that is involved in the statement are optional. They can be left out if we can make use of knowledge about the concepts of an angle and measure, and how they are related.

Statistical models based on bags of words are able to capture the relative relevance of different words for the sentence meaning, and so model the fact that some words are less necessary than others in order to get the right classification. There is no guarantee however that they will be able to capture accurately all possible cases. For instance in example (26) c) and d) one can see that '`equal`' is not necessary. In this context the predicate '`be`' means the same thing as '`be equal`'. Since measures are extensional concepts, identity and equality on them mean the same thing. In case of intensional concepts however their meanings are different:

(27) a) Angle BAC is angle DAE.

b) Angle BAC is equal to angle DAE.

Here (27) a) means that the two angles are actually the same (possible if points A, B, D are collinear, and points A, C, E are also collinear), while (27) b) means that only the measures of the two angles are equal, but the angles can still be different objects. Such constraints are harder to express with the right degree of generality in a system that does not rely on a logic model of the domain of discourse.

3.1.2.6. Anaphora

The presence of anaphora in natural language is yet another phenomenon that can lead to a significant difference in the set of words used to realize a given semantic content. Anaphora can come in various forms, the most common ones being pronouns and definite descriptions. Once the same referent of the anaphora is identified, these different forms result in the same semantic content. The anaphors can occur in different syntactic patterns, as shown by the examples below:

(28) a) The angles' sum is 90, because the angles are complementary.

b) The angles' sum is 90, because they are complementary.

(29) a) If two angles are complementary, then the sum of these angles' measures is 90 degrees.

b) If two angles are complementary, then the sum of their measures is 90 degrees.

Without a mechanism to accurately resolve the references, various approaches will have problems with identifying these forms as being semantically equivalent. The reference resolution mechanism can and should be based on a syntactic approach, since English exhibits various syntactic restrictions on positions where the two elements (the anaphor and the antecedent) can be placed with respect to one another, as expressed by various versions of Binding Theory (Pollard & Sag, 1994; Bresnan, 2001). However a syntactic approach alone cannot solve all cases, as shown in section 3.2.5 below.

3.1.3. Combinations

All the phenomena discussed so far can combine to result is a large variety of sentence forms that all have the same meaning. For illustration here are a few examples taken from our evaluation corpus that all express the same geometry theorem, about the measures of angles formed by other angles. Being able to accurately identify when such sentences are equivalent and when not is a difficult job without using extensive knowledge about the domain of discourse and the concepts involved.

(30) a) An angle formed by adjacent angles is equal to the sum of those angles.

b) The measure of an angle formed by other angles is equal to the sum of the measures of those angles.

c) The measure of an angle formed by interior adjacent angles is equal to the sum of the measures of those adjacent angles.

d) An angle's measure is equal to the sum of the two adjacent angles that form it.

e) The sum of the measures of two adjacent angles is equal to the measure of the angle formed by the two angles.

f) The measure of an angle formed by two adjacent angles is equal to the sum of the measures of the two angles.

g) If adjacent angles form an angle, its measure is their sum.

h) When an angle is formed by adjacent angles, its measure is equal to the sum of those angles.

3.2. Other problems that require semantic solutions

There are a number of other linguistic phenomena that have a strong influence on determining the meaning of natural language. Among the most wide-spread are plurals, structural ambiguity, metonymy, and anaphora. They are discussed in the following sections.

3.2.1. Plurals: Distributive vs. collective

Plurals pose various problems for a symbolic natural language model. One of the most important ones is to determine when the meaning of surrounding text applies to each element in a collection, and when it applies to the collection as a whole.

(31) a) Vertical angles formed by intersecting lines are congruent.

b) A pair of vertical angles formed by intersecting lines are congruent.

Example (31) b) above taken from our corpus seems to be ungrammatical. Whereas the subject is singular, the verb is in plural form. Besides, it does not seem to make sense, because sets themselves (of which 'pair' is a subclass) cannot be congruent. However, the sentence has a valid meaning, and that meaning is the same as in example (31) a). The subject represents a set of objects, while the predicate expresses a relation among the objects that are the elements of the set. In absence of knowledge about the concepts involved in the sentence, even if the number agreement constraint between subject and verb is relaxed, a syntactic approach can only build a structure where the set itself is congruent, not the angles that are elements of the set. A logic-based approach would be able to choose, in case of subjects that represent sets, based on the semantic features of

the properties involved ('congruent'), whether to assert the property on the set itself or on its elements.

(32) a) The angles in a triangle are 180 degrees.

b) The angles in a triangle are 60 degrees.

The problem with example (32) is that it is not clear whether the property is asserted about each of the angles in the set denoted by the plural subject or about the sum of their measures. That is, is this a distributive property or a collective property? It seems reasonable to consider that sentence (32) a) is about the sum of the measures, while (32) b) is about each measure. However syntactic only approaches cannot distinguish between such cases, since their syntactic form is identical. Knowledge about the semantic properties of concepts allows a semantic-based approach to consider various solutions. A system can build the distributive reading whenever semantic constraints allow the property to be asserted on the elements of the referenced set and build the collective reading whenever the set of elements forms a structure that has the asserted property well defined for it. A different solution could involve making a choice based on plausible ranges of values. Ambiguities that remain are also ideal candidates for tutorial dialog. For example, the cognitive tutor can respond to (32) a) with: 'Is every angle in a triangle equal to 180 degrees?'

3.2.2. Syntactic ambiguity: Prepositional phrase attachment

Syntactic ambiguity in many cases does not reflect semantic ambiguity. One of the most widespread cases of structural ambiguity in English is prepositional phrase attachment (see Altmann & Steedman, 1988). That is, following only the rules of the grammar, in many cases a prepositional phrase could be an adjunct or argument to several different preceding components. A deeper look at those alternative attachments reveals that most of them can be discarded because they do not result in a meaningful sentence. However, in absence of detailed knowledge about the meaning of the words in the sentence and their possible interactions, a Natural Language Understanding approach would not be able to disambiguate among them.

(33) The sum of the measures of the three interior angles in a triangle is equal to 180 degrees.

Example (33) contains these three prepositional phrases: 'of the measures', 'of the three interior angles', and 'in a triangle'. Whereas the first phrase can only be attached to one place, the noun 'sum', the second phrase can be attached to two places: 'sum' or 'measures'. This case can be solved by using the syntactic information that 'sum' and 'measures' both take a prepositional 'of' phrase as argument (since they express relations), and they can only have one such argument, so the only place left for the attachment of the phrase 'of the three interior angles' is to 'measures'. However, the third phrase can be attached to three different places: 'sum', 'measures', or 'angles'. And while empirically/psychologically motivated preferences could be considered (like preferring the closest attachment point to more distant ones), there is no syntactic constraint allowing one choice over the others in all cases. By combining knowledge about the concepts of 'sum', 'measure', and 'angle' with that on 'triangle' and possible relationships expressed by the preposition 'in' one can show

that sums of measures in a triangle can only be done over some elements related to that triangle, thus allowing the choice of '`angles`' as the attachment point for '`in a triangle`' (or alternatively allowing to derive the same semantic structure from all attachment places).

3.2.3. More structural ambiguity: Noun-noun compounds

Another widespread case where structural ambiguity does not reflect semantic ambiguity is that of noun-noun compounds. Because of the specific feature of English that allows nouns to be used as adjectives that modify other nouns, sequences of nouns combined with adjectives can easily result in many different syntactic structures, most of which do not make sense. A few examples are given below:

(34) a) `Linear pair angles sum is 180.`

b) `Isosceles triangle base angles are congruent.`

In (34) a) '`sum`' could modify any of '`angles`', '`pair`', or '`linear`' and only knowledge about what kind of elements can be summed can choose the right structure. Similarly, the correct relation between '`angles`' and the two preceding elements (that the '`angles`' are the elements of the '`pair`') can only A more complex example is given in (34) b), where while '`base`' can only modify '`angles`', '`triangle`' could modify either '`base`' or '`angles`' and '`isosceles`' has three choices: '`triangle`', '`base`', and '`angles`'. These choices lead to four different syntactic structures. Semantic restrictions allow a system to select '`isosceles`' to modify '`triangle`', since it is a property that only applies to triangles. Although semantic information cannot select between '`triangle base`' and '`triangle angles`' since both have valid meaning, it would allow a system to identify both as being semantically equivalent, since both actually mean the same as '`base angles of isosceles triangles`'.

3.2.4. Apparent semantic ill-formedness: Semantic structure does not follow syntactic structure: metonymy, sets

It is often the case that the semantic structure of a sentence as determined by the nature of the concepts involved does not directly follow its syntactic structure. One such case is that of metonymies. Another one involves talking about sets of objects.

Metonymy is "a means by which one entity stands for another" (Fass, 1988). Or more precisely, it is the phenomenon of imprecise reference whereby a semantically incorrect concept is used as a shortcut for the correct concept. Let's consider an example:

(35) a) The sum of a linear pair of angles is 180.

b) The sum of the measures of a linear pair of angles is 180.

c) The sum of the measures of a pair of linear angles is 180.

d) The sum of the measures of the linear angles in a pair is 180.

e) The sum of the measures of the linear angles that are elements of a pair is 180.

In sentence (35) a) above, it is technically incorrect to add angles, since we can only add numerical quantities, like measures. Thus, '`a linear pair of angles`' stands for '`the measures of a linear pair of angles`'. Knowledge about the concepts of '`adding`' and '`angles`' will provide the necessary information that angles are added through their measures, thus allowing the reconstruction of the correct semantic structure. The problem is not so much allowing a semantic structure where angles are added instead of measures. It rather comes from the requirement to recognize that this structure is logically equivalent to adding measures of angles, so the two structures will have the same semantic representation. The problem is composed here with the problem of collective vs. distributive reading with respect to sets of objects, as discussed in section 3.2.1. Thus, in (35) b) a '`pair`' cannot be '`linear`', since '`linear`' is a relation that can be applied only to geometry objects, like '`angles`'. Then '`a linear pair of angles`' actually stands for '`a pair of linear angles`'. But then in (35) c) the '`measures`' cannot be measures of '`a pair`', because a pair can only have one measure, its cardinality, and that is 2. The '`measures`' are of course of '`the angles in a pair`', so the complete phrase would be that given in sentence (35) d). Actually even this structure needs further refinement to specify what kind of relation the generic '`in`' stands for. The fact that it is followed by a set allows us to conclude the angles are the elements of that set, leading to sentence (35) e).

Metonymy is used in 392 sentences of our corpus (about 56%), the most widespread case being of angles used instead of their measures.

3.2.5. Reference resolution

Section 3.1.2.6 discussed how the presence of anaphora results in cases where sentences with different sets of words are equivalent semantically. This problem leads to the necessity to have an accurate reference resolution mechanism. Without solving the references accurately, an approach to NLU would not be able to build the right semantic representation for the sentence, and thus it would fail to recognize the semantic equivalence. Our corpus of 700 sentences contains 141 sentences (about 20%) with anaphora in all forms, reciprocals, pronouns, or definite nouns.

Finding the right referent for an anaphor is not always easy. Syntactic criteria can help with disambiguating among candidates, but there are cases where they cannot lead to a unique candidate. Adding semantic constraints to the solution can increase the accuracy considerably. This is true especially in a domain like geometry, where the referent

candidates are usually various geometry concepts with very different semantic properties. An example is given below:

(36) If the lengths of two sides of a triangle are equal, then the measures of the angles opposite them will also be equal.

In this example the pronoun 'them' in the main clause is used as a collective reference to the discourse referent denoted by 'two sides' in the conditional clause. There are however several different candidate referents for binding 'them' besides the right one: 'the lengths', 'a triangle', 'the measures', and 'the angles'. A syntactic approach would eliminate 'the angles' based on the syntactic constraint that a pronoun has to be free in its local domain (Pollard & Sag, 1994). However, both 'the lengths' and 'the measures' satisfy all syntactic constraints, and can only be disambiguated using knowledge that geometry objects cannot oppose measures. While in theory 'a triangle' could be ruled out based on mismatched number information, the necessity of robustness to understand users who use number unreliably makes using this constraint more difficult. Besides the sentence could have had 'triangles' instead of 'a triangle' thus making this constraint inoperative. Again the same semantic constraint used before says that angles cannot be opposite triangles either, and thus strengthens the choice.

The problem of anaphora resolution can be further complicated by the presence of sets of objects that can be referenced collectively, as in the examples below taken from our corpus:

(37) a) The sum of angle VEC and angle TEO is 180 degrees because they are supplementary.

b) Angle LGH and TGH are supplementary, and these angles are linear angles so they must be supplementary and their measures sum to 180 degrees.

In these examples 'they' in the second or third clause does not have a good referent in the first clause in a syntactic only approach. A semantic approach is needed to recognize the fact that the two angles in the first clause can form a set, and then that elements of this set can be referred collectively by the pronoun 'they'. Moreover, the choice is validated by the fact that angles are the right kind of objects that can be summed up, through their measures, as discussed above.

Chapter 4 Alternative Approaches

A number of Intelligent Tutoring Systems developed so far are using different techniques for dealing with student input expressed in Natural Language. This chapter analyses the most representative approaches, as well as a few complementary approaches from the Machine Translation domain.

4.1. Statistical based approach - Autotutor

Autotutor (Wiemer-Hastings & al, 1999) is a tutor that relies exclusively on natural language dialog to tutor college students in the domain of computer literacy. It takes a corpus-based, statistical approach to natural language processing, called Latent Semantic Analysis. This approach is part of the "bag of words" class of approaches, where the relevant pieces of information in the text are the words, their sequence or structure in the sentence being ignored. The underlying idea is that the aggregate of all the word contexts in which a word appears determines the similarity of meaning of words to each other (Landauer & al, 1998).

Thus, the tutor starts with a curriculum script that contains a representation of the questions or problems that the tutor can handle in its domain of expertise. For each topic in the domain, it has a number of questions or problems graded according to their difficulty level. Each such problem/question is accompanied by a lengthy complete "ideal" answer, a breakdown of that complete answer into a set of good answers that cover parts of the complete answer, a set of additional good answers, a set of bad answers, a set of questions that the students might ask together with appropriate answers, and a summary.

Each of the previously mentioned items is used as input in the statistical training phase of LSA. The corpus also includes additional information from textbooks and articles about computer literacy. All this information is divided at the level of paragraphs, which constitute separate text inputs for the trainer.

The basic claim of the approach is that terms that occur in similar contexts carry similar semantic information. Thus, LSA computes a co-occurrence matrix of terms and texts, the cells representing the number of times each term occurs in each text. A term is taken to be any word that occurs in more than one such text. A log entropy weighting is performed on this matrix to emphasize the difference between the frequency of occurrence for a term in a particular text and its frequency of occurrence across texts. The matrix is then reduced to a number of dimensions by a type of principle components

analysis called singular value decomposition. The result is a set of weights and a set of vectors, one for each term, and one for each text.

Then the vector for a text is computed as the normalized sum of the vectors of the terms in the text. The distance between two vectors, computed as the geometric cosine, is interpreted as the semantic distance between the terms or the texts the vectors correspond to.

The results of the training process are used during the tutoring sessions to evaluate student responses. A normalized vector is computed from the student's response. This vector is then compared with text vectors for some of the curriculum script items for the current topic. Two measures are computed: a) completeness, as a percentage of the aspects of a complete answer that match the student's response; b) compatibility, as a percentage of the student's response that match some aspect of the complete answer. A match is defined by the cosine semantic distance being above a certain threshold. Optimal parameters of the process were chosen by comparing the correlation between the system's results and a panel of 3 human graders.

4.1.1. Limitations of statistical approaches

A common problem with all statistical approaches is the inability to finely control the outcome of the analysis. In case a specific problem is detected with the system's response, all one can do is train it further with specific examples, in the hope of getting better results.

The LSA approach seems to be more suitable for analyzing larger pieces of text, like full paragraphs. In our case most responses are only one sentence long, or even less, which might not give much material for computing a relevant vector.

Another problem is that the LSA approach as used in the Autotutor only gives a score of how well the response matched the expected complete answer, without being able to tell exactly what was wrong with the response. By comparing the student's response to a set of partial answers, LSA could also be used as a classification device, to try to give information about what is missing from the response. Such use could require large amounts of labeled data to achieve reasonable performance.

Even if used as a classifier, the reliability of the classification would be questionable for at least two reasons. First, being a "bag of words" approach, it will miss differences between responses where the same set of words is used, but with a different syntactic/semantic structure. Such an example was given in (13), repeated here:

(38) a) Angles opposite to congruent sides in a triangle are congruent.

b) Sides in a triangle opposite to congruent angles are congruent.

Second, many times the difference between a completely good explanation and a partial explanation could consist of just one word, as in the example below. In the context of a long sentence using the same set of words, a one-word difference might be hard to distinguish by an LSA approach.

(39) a) The measure of an angle formed by other adjacent angles is equal to the sum of the measures of those angles.

b) The measure of an angle formed by other angles is equal to the sum of the measures of those angles.

4.2. Semantic grammars

4.2.1. Sophie

Sophie (Brown & al, 1982) is a sequence of three tutoring systems used in teaching electronics to Air Force personnel. In its third version, the system consists of three major expert modules: the electronic expert, the troubleshooter, and the coach. These expert systems are used together in a variety of scenarios to solve electronics problems, model student's knowledge and inference processes, and provide explanations about the tutor's reasoning.

Sophie's natural language interface is designed to understand student's sentences about troubleshooting an electronic circuit. It is based on two techniques. First, it incorporates the domain semantics into the parsing process using semantic grammars. Second, it uses a dialogue mechanism to handle constructs that arise in conversation.

The natural language understander's target is to translate the users queries from natural language into a functional representation consisting of objects of the electronic circuit and functions operating on them. Knowledge of natural language is encoded into the semantic grammar in the form of grammar rules that give for each function or object all possible ways of expressing it in terms of other constituent concepts. For example, <measurement> expresses all of the ways in which a student can refer to a measurable quantity and also supply its required arguments. Rules have associated with them methods for building the meaning of their concepts from the meanings of the constituent concepts. This allows the semantic interpretation to proceed in parallel with the recognition process.

4.2.2. Nespole!

Nespole! (Lavie & al, 2001) is a system designed to provide fully functional speech-to-speech machine translation in a real-world setting of common users involved in e-commerce applications. Nespole! takes an interlingua-based approach with a relatively shallow task-oriented interlingua representation. The system implements a client-agent architecture, where communication is facilitated by a dedicated module.

For each language the system provides a language-specific server that consists of an analysis chain and a generation chain. The analysis chain consists of a speech recognition module and a text analysis module. Similarly the generation chain consists of a text generation module and a speech synthesis module.

Nespole! defines an Interchange Format Interlingua representation consisting of mainly four elements: a speaker tag, a speech act, and optional sequence of concepts, and an optional set of arguments. The speech act and the concept sequence are called the domain

action. Nespole! uses a hybrid analysis approach. In the first step input utterances are parsed with four phrase-level semantic grammars (argument grammar, pseudo-argument grammar, cross-domain grammar and shared grammar) using a robust parser. In the second step the input is segmented into semantic dialogue units using a statistical model. In the third stage the input is passed to a automatic domain action classifier whose purpose is to identify the domain action for each semantic dialogue unit.

4.2.3. Limitations of the semantic grammars approach

Semantic grammars represent a means to combine semantics with syntax knowledge. A semantic grammar rule specifies a syntactic pattern for expressing a given concept, but whose components are also other concepts. Thus, the grammar represents a cross product between syntax and semantics.

This fact provides the formalism with a degree of flexibility not present in models based on separation of syntax and semantics. Any time the need to recognize a new pattern arises, one can write a new grammar rule for that specific pattern.

However, this flexibility comes at a price. The size of a semantic grammar grows approximately proportional with the product of the new domain knowledge and the syntactic patterns involved.

A second problem is that the most of the semantic grammar developed for a domain is specific to that domain, and thus is not extensible to new domains, even if the same language is used. The problem could be alleviated by the fact that parts of the semantic grammar could be reusable, specifically those corresponding to the "upper level" knowledge, which is more or less domain-independent.

4.3. Finite state syntax approach - CIRCSIM-Tutor

CIRCSIM-Tutor (Glass, 2000) is an intelligent tutoring system designed to tutor first-year medical students on the baroreceptor reflex, a mechanism for blood pressure regulation in the human body. It performs its tutoring task exclusively through natural language dialogue. The tutor asks questions to students and interprets their answers.

In the course of the tutoring dialogue, the system presents the student with a description of a perturbation that disturbs blood pressure in the human body. The student is then asked to predict the direct effect on seven physiological variables, and how they will change.

The natural language interpretation capabilities of the system are limited. All questions admit short one- or two-word answers. The system processes natural language inputs by searching through the sequence of words for the relevant pieces of information and ignoring the rest.

The linguistic processing of text consists of 4 steps: lexicon lookup, spelling correction, recognition by finite state transducers, and lookup in concept ontologies. Lexicon lookup basically retrieves the stem form of the word and its part of speech. The spelling correction is invoked when lexicon lookup fails.

Recognition is performed by a cascade of finite-state transducers. Each transducer looks for a specific kind of answers, like neural mechanism or parameter change. Most of the time, transducers perform simple keyword lookup. In a number of cases, transducers also perform a limited amount of syntax recognition, like simple prepositional phrases.

The lookup in concept ontologies tries to determine the appropriateness of the answer in the context of the question. Thus, the system has separate ontologies for each question. Each such ontology gives a simple map of acceptable answers for that particular question. The meaning tokens found in the parsed input are thus classified according to these maps. The results are matched against the question by ad hoc code, which produces a representation of the answer.

This understander is a replacement in the system for a previous understander that was performing more syntactic parsing. Among the reasons quoted for the replacement were poor handling of extragrammaticality (like lack of capability to skip words), and difficulty in extracting the meaning from a variety of syntactic forms it could be expressed as.

4.3.1. Limitations of the finite state syntax approach

The requirements for our tutor make such an approach unsuitable. Geometry theorems are usually full sentences with a precise logical structure, so a one- or two-word answer is not acceptable. That could be enough to identify the name of the theorem in question, but not enough to actually express its full content. Thus our approach needs to perform a full syntactic analysis of the student's answer and try to reliably determine its full semantic content.

We face the same problems that they quote for replacing their previous understander. We cope with the variety of syntactic forms for the same semantic content by using an extensive ontology of the domain of discourse, implemented in a Description Logic framework. The ontology, together with a compositional semantics approach for the mapping from syntax to meaning, gives us the right semantic concepts and relations that underlie various syntactic structures.

4.4. Deep syntactic/frame-based semantic approach

4.4.1. Atlas-Andes tutor

Atlas-Andes (Rosé, 2000) is a tutor that adds a dialogue capability to a previous Andes physics tutor. Thus the new tutor replaces the sequences of hints that the Andes tutor was using to help students cope with difficulties in understanding with generated natural language subdialogues. The task of Carmel, the natural language understander in Atlas, is to extract relevant information from student explanations and other natural language input to pass back to the planner.

Carmel takes a syntactic/semantic approach very similar to ours. Thus, after spelling correction and lexicon lookup, the text is parsed using a robust parser and a broad coverage English grammar. The parsing process also constructs a meaning representation

specification. A repair process is called for cases that were not covered by the grammar. The repair process relies solely on semantic constraints in trying to assemble pieces of a fragmentary parse.

The LCFlex syntactic parser produces feature structures that associate deep syntactic roles with phrases in the sentence. The lexicon provides the correspondence of the syntactic roles to semantic arguments. The semantic formalism used is an ad-hoc frame based language that allows the specification of a hierarchy of semantic types, together with semantic restrictions of associated roles. This specification is compiled into semantic constructor functions for efficiency reasons.

The Autosem semantic interpretation module takes a representation of the domain of discourse as a set of constructor functions, and uses it to build the semantic representation of the sentence in the same feature structure as the syntactic representation, using extensions of the unification formalism. The same module is then used in the repair process, if necessary, to assemble semantic fragments into a single semantic structure, while observing domain specific semantic restrictions.

4.4.2. Limitations of Atlas-Andes system

The approach is more suitable than alternatives above for capturing the deep meaning of student's NL sentences, one of the main requirements in our application. However, the absence of a logic system behind the semantic formalism limits its power. The only kind of inference that the system is able to perform is to percolate semantic restrictions down in the inheritance hierarchy. Thus, the system has difficulties with recognizing widely different surface realizations of the same semantic content as meaning the same. Such cases require deduction capabilities that would infer new properties and relations based on existing sets of semantic constraints.

One such example, given in (23) above, is that of using definitions instead of named concepts:

(40) a) In a triangle with two congruent sides the base angles are congruent.

b) In an isosceles triangle the base angles are congruent.

In order to recognize the two sentences as meaning the same thing, the system should be able to recognize that '`a triangle with two congruent sides`' represents a definition of an '`isosceles triangle`', and thus that the two sentences are logically equivalent. In absence of such capabilities, the system would still have to keep lists of equivalent semantic forms for each structure that it needs to recognize.

A related problem comes from the fact that the approach uses for semantic representation the same tree-based pseudo-unification formalism used for syntax representation. This formalism lacks the flexibility necessary in cases where there is a need to look at the same structure from different points of view, again with negative consequences on the system's ability to recognize semantic equivalence. As an example, we can take two of the sentences from example (30).

(41) a) An angle formed by two adjacent angles is equal to the sum of those angles.

b) The sum of two adjacent angles is equal to the angle formed by those angles.

These two sentences have the same meaning, but in the first sentence the relation is presented from the point of view of the whole angle towards its parts, while in the second sentence it is the other way around.

Besides, the example above also needs a strong reference resolution mechanism, to be able to identify the referential description '`those angles`' as referring to the same entities as '`the two adjacent angles`'.

4.4.3. KANT/KANTOO machine translation system

The KANT system (Knowledge-based Accurate Natural-language Translation) and its object-oriented successor KANTOO are machine translation systems for automatic translation of documentation in technical domains. (Nyberg & al, 1998). KANT supports the translation process by providing a number of modules and/or tools for the documentation author.

Thus (Nyberg & Mitamura, 2000), the analyzer performs tokenization, morphological processing, lexical lookup, syntactic parsing with a unification grammar, and semantic interpretation into an Interlingua form. It supports the use of controlled language with explicit conformance checking using the machine translation grammar. As part of the analysis process, it also supports interactive disambiguation at the lexical or structural level. The generator performs lexical selection, structural mapping, syntactic generation, and morphological realization for a particular target language. The lexicon maintenance tool and the knowledge maintenance tool allow the author to edit the lexicon entries, the syntactic grammars, and the translation rules.

4.4.4. Challenges for the KANT system

Similarly to the Atlas/Andes system, the KANT system relies mostly on syntactic parsing in its analysis of input language. The Interlingua provides a certain degree of surface language independence, but the need for structural mapping rules demonstrates that it cannot achieve complete independence. In machine translation total independence of the input language might not even be desirable, since the target language needs to keep the general line of the source language. Thus the KANT system is able to achieve high-precision translation, but not high-precision understanding, due to an absence of any reasoning capabilities. The system also benefits from the possibility to control the input language, both by limiting the vocabulary and the allowed word senses, and by ruling out the most difficult sentence structures. Such control is not possible in the tutoring domain.

4.5. Deep-syntactic/logic-based semantic approach – Gemini

Gemini (Downing & al, 1993) is a natural language understanding system developed for spoken language applications. Gemini combines a syntactic chart parsing process using a

conventional unification grammar with two rule-based recognition modules: one for gluing together the fragments found during the parsing phase, and another for recognizing and eliminating disfluencies.

Processing starts in Gemini when syntactic, semantic, and lexical rules are applied by a bottom-up all-paths constituent parser. Then a second utterance parser is used to apply a second set of syntactic and semantic rules that are required to span the entire utterance. If no semantically acceptable edges are found, a component to recognize and correct grammatical disfluencies is applied. When an acceptable interpretation is found, a set of parse preferences is used to choose a single best interpretation from the chart. Quantifier scoping rules are applied o this interpretation to produce the final logical form.

Gemini maintains a firm separation between the language and domain-specific portions of the system, and the underlying infrastructure and execution strategies. Gemini includes a midsized constituent grammar, a small utterance grammar, and a lexicon, all written in the unification formalism from the Core Language Engine (CLE) (Alshawi, 1992). It used typed unification, but with no type inheritance.

The lexicon includes base forms, lexical templates, morphological rules, and type and feature default specifications. The constituent grammar consists of syntactic rules and semantic rules. The syntactic rules specify the components and the values for the feature structure. The semantic rules enforce semantic constraints and associate a logical form with each constituent. The parser uses subsumption checking to reduce the chart size. The syntactic and semantic processing is interleaved, at each step all possible sortal constrains are imposed on the logical form.

The utterance grammar specifies ways of combining the categories found by the constituent parser into a full-sentence parse, by stating constraints on the position and sequence of constituents in the sentence. In case no acceptable interpretation is found for the complete utterance, a repair mechanism is applied to correct the input string by deleting a number of words. The candidates are ordered by the fewest deleted words, and the first one that can be given an interpretation is accepted.

In case several parse trees are acceptable, a rule-based preference mechanism is applied to choose the best interpretation. The preference mechanism uses rules like minimal attachment and right association (Kimball, 1973). Finally, a set of quantifier scoping preference rules is applied on the chosen logical form to produce the final form.

4.5.1. Limitations of the Gemini system

The Gemini system comes closer to the approach proposed in this thesis, in the fact that the final result is a logical form. Unlike in our approach, the logical form is a classical quantified logic formula. The system also implements a limited model of the domain of discourse, which, even if applied on a logical form, is pretty similar to the Atlas/Andes system in terms of representation power, thus being subject to the same limitations.

Chapter 5 System Description

The adopted architecture lies within the class of knowledge based approaches. It consists mainly of two subsystems: the syntactic processing system and the semantic processing system. These two subsystems interact during the NLU process, but keep separate structures and use different types of knowledge bases. They work together to build syntactic and semantic representations for natural language sentences. The semantic representations are then classified according to a hierarchy of classes and relevant classes are returned to the tutor.

5.1. System architecture

The system's overall architecture is presented in Figure 5-1. The syntactic processing system uses as its main engine an active chart parser called LCFlex (Rosé & Lavie, 1999). The semantic processing system bases its action on a Description Logic system called Loom (MacGregor, 1991).

The *interface module* is responsible for connecting the NLU subsystem to the tutor itself. It works asynchronously to the NLU system. On the syntactic processing side, the interface module takes the input sentence from the tutor, and after performing some preprocessing and spelling checking, it passes it as a sequence of words to the chart parser, one by one. It does that in real time, while the input string is still being typed and/or edited. It then waits for the parser to finish and passes the resulting classifications back to the tutor.

The *chart parser* (Kay, 1986) is the main engine of the system. It uses linguistic knowledge about the target natural language from the *unification grammar* (Shieber & al, 1983) and the *lexicon*. The parser takes words of a sentence one by one and parses them according to rules in the unification grammar. During the process, it builds *feature structures* for each phrase successfully recognized. These feature structures store lexical, syntactic, and semantic properties of corresponding words and phrases. The parser uses an *active chart* that serves as a storage area for all valid phrases that could be built from the word sequence it received up to each point in the process.

The parser calls the *feature structure unifier* in order to process restrictions attached to grammar rules. The feature structure unifier ensures that these restrictions, expressed in the form of equations, are satisfied. The equations operate over the feature structures corresponding to different components of the phrase. They either check for compatibility/identity between elements/sub-structures of these feature structures, or build new sub-structures based on existing elements in the other feature structures.

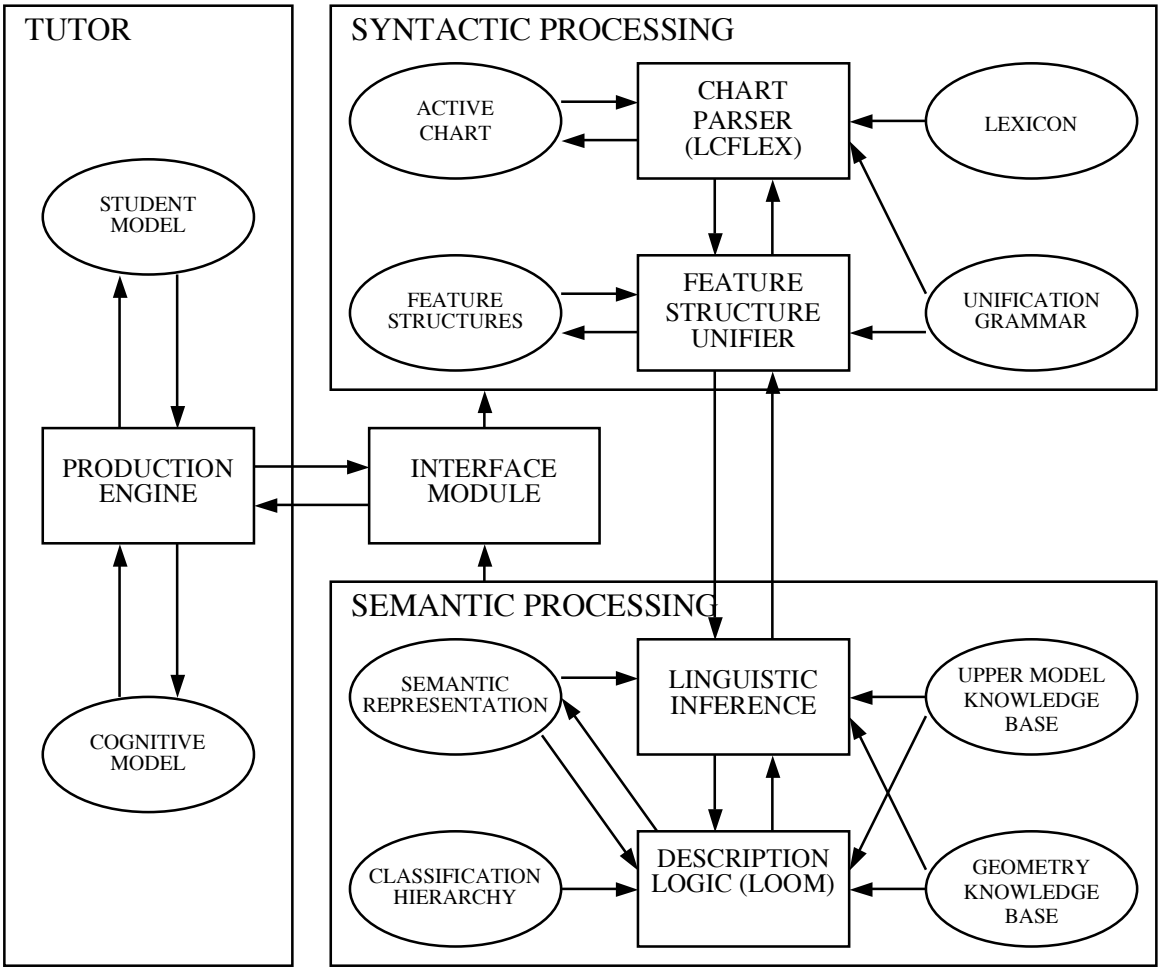

Figure 5-1. System architecture

The feature structure unifier also ensures the interaction between the syntactic and the semantic systems. Some of these equations, instead of working on the features structures, are directives to the *Description Logic system* (Baader & al, 2003). The Description Logic system relies on a model of the domain of discourse, encoded as concepts, relations, and productions in the *knowledge base*. Both concepts and relations stand for predicates in the underlying logic. Productions perform additional inferences that are harder to encode into concepts and/or relations.

The interaction between the feature structure unifier and the Description Logic system is mediated by the *linguistic inference* module. This module is responsible for performing semantic processing that is specific to natural language understanding, like resolving metonymies and resolving references.

Based on this knowledge base, the system creates a model-theoretic *semantic representation* for the sentence as a set of instances of various concepts connected through various relations. An instance corresponds to a discourse referent in the sentence. The logic system also ensures that the semantic representation is coherent semantically, that is that it observes all semantic restrictions between the concepts involved in the sentence. Links to this representation are stored in the feature structures. The logic system

then uses a classifier to evaluate the semantic representation against a *classification hierarchy* of valid representations for geometry theorems. The results of the classification are passed back to the tutor.

5.2. Syntactic processing

5.2.1. Chart parser

The parser used in the system is LCFlex, a left-corner active-chart parser developed at University of Pittsburgh (Rosé & Lavie, 1999). The parser works in tandem with a feature structure unifier that originates in a previous system, GLR*, developed at Carnegie Mellon University (Tomita, 1988; Lavie, 1995). Thus the grammar rules recognized by the syntactic processing system have two parts: a context-free part, and a unification part.

The context-free parts of the rules look as in example (42) below. They consist of a left-hand side category (the mother category) followed by a sequence of right-hand side categories (the daughters). Its meaning is that a sequence of words and/or phrases identified as being of the categories on the right-hand side will form a new phrase whose category is the one on the left-hand side of the rule. A number of lexical-level rules are also needed to generate the basic lexical categories, like Det, Adj, Num, N, P, V in the example below. They can be generated based on information about individual words in the lexicon.

```
(42)  (<S> ==> (<NP> <VP>))
      (<NP> ==> (<Det> <N1>))
      (<NP> ==> (<N1>))
      (<N1> ==> (<Adj> <N1>))
      (<N1> ==> (<Num> <N1>))
      (<N1> ==> (<N1> <PP>))
      (<N1> ==> (<N>))
      (<PP> ==> (<P> <NP>))
      (<VP> ==> (<V> <NP>))
```

A left-corner parser (Rosenkrantz & Lewis, 1970) is a type of bottom-up parser with additional top-down (or left-corner) prediction. Thus it starts with individual words and it aggregates them into larger phrase structures allowed by the rules of the grammar. The active chart parser avoids parsing the same phrase twice by storing the partially parsed phrases in a chart. Thus the chart serves mainly as a memoization mechanism that reduces the complexity of the algorithm. The left-corner prediction works by validating the phrase structures the parser builds at each stage based on grammar rules predicted in a top-down way, starting with the top-level symbol at the beginning of the sentence.

5.2.2. Active chart

The chart is structured as a directed acyclic graph built around a backbone of nodes corresponding to positions between words in the sentence. At any point during the parsing process it contains two types of arcs (Kay, 1986):

- passive – those corresponding to completely identified phrases, that is phrases modeled by rules whose right-hand side sequences have been fully parsed;
- active – arcs corresponding to phrases that have been only partially identified thus far and wait for new elements to be added to become complete.

The LCFlex parser takes words of the input sentence one by one and creates an active arc in the chart for each of them. It then starts from left to right, going through the chart nodes one by one. At each step it repeatedly does two operations, whenever possible: create new active arcs for new rules that could model the current word sequence (chosen accordingly to the left corner prediction), and/or combine two successive arcs (first active and second passive) into a new arc. In the second case, the two arcs are taken such that the category of the passive arc matches the next element needed for completion of the right-hand side of the rule on the active arc. Thus at each point in time the chart keeps all possible partial and complete phrase structures parsed up to that point.

Let's take the sentence in example (43). The chart produced by a left-corner active-chart parser working with the simplified context-free grammar in example (42), and the extra lexical-level rules given in example (44), can be seen in Figure 5-2.

(43) The measure of a right angle is 90 degrees.

```
(44) (<Det> ==> ("the"))
     (<N> ==> ("measure"))
     (<P> ==> ("of"))
     (<Det> ==> ("a"))
     (<Adj> ==> ("right"))
     (<N> ==> ("angle"))
     (<V> ==> ("is"))
     (<Num> ==> ("90"))
     (<N> ==> ("degrees"))
```

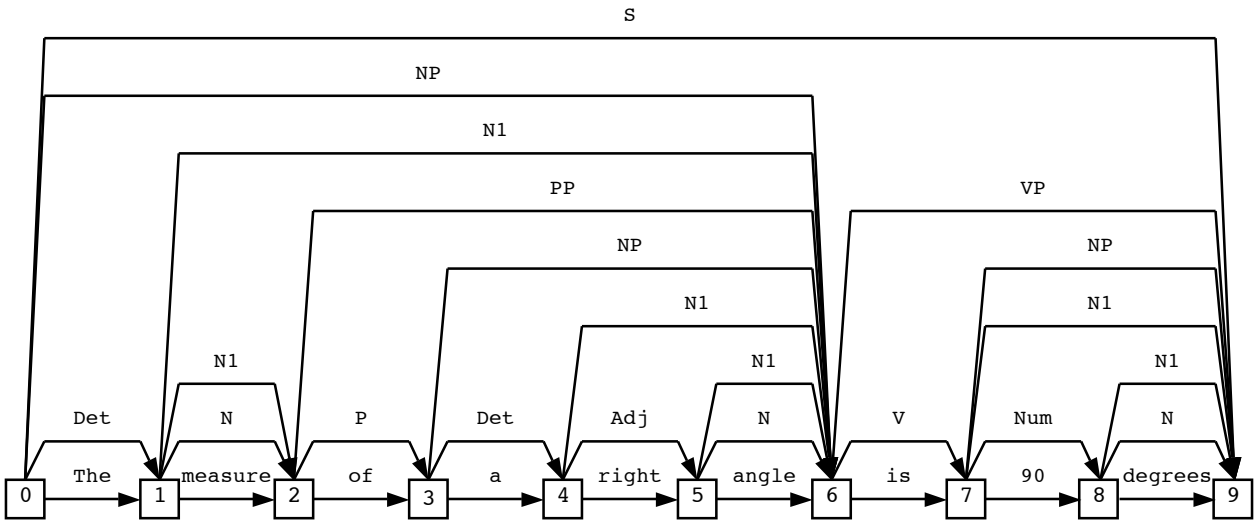

Figure 5-2. Example of active chart for sentence "The measure of a right angle is 90 degrees."

5.2.3. Unification grammar

The grammar formalism used consists of context free rules accompanied by equations. The equations serve two purposes:

- to build feature structures associated with recognized phrases in the sentence;
- to specify restrictions among the feature structures corresponding to the elements on the right-hand side of the rule.

These restrictions are verified by the feature structure unifier described in section 5.2.5.

As shown in example (45) below, each equation specifies equality between two paths. The two paths could belong to the same or to two different features structures. The paths start with an `xn` identifier, which specifies the feature structure the path is part of, based on its relative order in the rule. Thus `x0` is the feature structure for the left-hand side of the rule, and `x1`, `x2`, … correspond to elements on the right-hand side.

The current grammar used in the system follows loosely the Lexical Functional Grammar theory (Bresnan, 2000). An example of how the grammar in example (42) could be augmented with highly simplified unification equations following LFG theory, which puts the elements together to build new feature structures, is shown in example (45).

```
(45)  (<S> ==> (<NP> <VP>)
           ((x0 = x2)
            ((x0 subject) = x1)))
      (<NP> ==> (<Det> <N1>)
            ((x0 = x2)
             ((x0 determiner) = x1)))
      (<NP> ==> (<N1>)
            ((x0 = x2)))
      (<N1> ==> (<Adj> <N1>)
            ((x0 = x2)
             ((x0 attribute) = x1)))
      (<N1> ==> (<Num> <N1>)
            ((x0 = x2)
             ((x0 numeral) = x1)))
      (<N1> ==> (<N1> <PP>)
            ((x0 = x1)
             ((x0 modifier) = x2)))
      (<N1> ==> (<N>)
            ((x0 = x1)))
      (<PP> ==> (<P> <NP>)
            ((x0 = x1)
             ((x0 object) = x2)))
      (<VP> ==> (<V> <NP>)
            ((x0 = x1)
             ((x0 object) = x2)))
```

The lexical-level rules could be built generically to just check for the appropriate part of speech for each word, which is specified in the lexicon. In example (46) below '`%`' stands for "any word".

```
(46)  (<Det> ==> (%)
              (((x1 cat) = det)
               (x0 = x1)))
      (<N> ==> (%)
           (((x1 cat) = n)
            (x0 = x1)))
      (<P> ==> (%)
           (((x1 cat) = p)
            (x0 = x1)))
      (<Adj> ==> (%)
              (((x1 cat) = adj)
               (x0 = x1)))
      (<V> ==> (%)
           (((x1 cat) = v)
            (x0 = x1)))
      (<Num> ==> (%)
              (((x1 cat) = num)
               (x0 = x1)))
      (<N> ==> (%)
           (((x1 cat) = unit)
            (x0 = x1)))
```

5.2.4. Lexicon

The lexicon stores information about words of the language. Each word has associated with it a lexical feature structure that specifies the main lexical, syntactic, and semantic characteristics of the respective word. For instance, for the words in example (43), the simplified lexical structures needed for the grammar above would be:

```
(47)  ((:word "the") (:cat det))
      ((:word "measure") (:cat n))
      ((:word "of") (:cat p))
      ((:word "a") (:cat det))
      ((:word "right") (:cat adj))
      ((:word "angle") (:cat n))
      ((:word "is") (:cat v) (:arguments ((subject object))))
      ((:word "90") (:cat num))
      ((:word "degrees") (:cat unit))
```

In general, only stem forms of words have an entry in the lexicon. Inflectional forms are processed by a lexical analyzer that combines knowledge about English morphology with information in the lexicon.

5.2.5. Feature structure unifier

The feature structure unifier is the process that takes unification equations and applies them onto feature structures. It starts by building an empty feature structure for each left-hand side non-terminal of a rule that is applied. It then takes equations one by one and applies them onto the specified feature structures. If all equations succeed, the grammar rule succeeds and the newly built feature structure is returned to the parser. If any of the equations fail, the entire rule fails and nothing is built.

As shown in section 5.2.3 an equation consists of equality between two paths. The equality specifies that the feature structures at the end of the paths must unify, in order for the equation to succeed. The LCFlex unifier uses a pseudo-unification formalism (Tomita, 1988). Under this formalism, if the two feature structures already exist, the unification process checks that their substructures unify, recursively. If one of the feature structures is empty, the unification will copy the existing substructure to the empty one.

For example, let's say we parse the sentence in example (43), using the simplified unification grammar in example (45) and (46). Right before applying the rule:

```
(48)  (<S> ==> (<NP> <VP>)
           ((x0 = x2)
            ((x0 subject) = x1)))
```

the unifier will have the feature structure for <NP>:

```
(49)  ((WORD "measure") (CAT N)
       (DETERMINER ((WORD "the") (CAT DET)))
       (MODIFIER
         ((WORD "of") (CAT P)
          (OBJECT
            ((WORD "angle") (CAT N)
             (DETERMINER ((WORD "a") (CAT DET)))
             (ATTRIBUTE ((WORD "right") (CAT ADJ)))))))))
```

and the feature structure for <VP>:

```
(50)  ((WORD "is") (CAT V)
       (SUBJECT)
       (OBJECT
         ((WORD "degree") (CAT UNIT)
          (NUMERAL ((WORD 90) (CAT NUM))))))
```

After applying rule (48), the unifier will build the feature structure for <S>:

```
(51)  ((WORD "is") (CAT V)
       (SUBJECT
         ((WORD "measure") (CAT N)
          (DETERMINER ((WORD "the") (CAT DET)))
          (MODIFIER
            ((WORD "of") (CAT P)
             (OBJECT
               ((WORD "angle") (CAT N)
                (DETERMINER ((WORD "a") (CAT DET)))
                (ATTRIBUTE ((WORD "right") (CAT ADJ)))))))))
       (OBJECT
         ((WORD "degree") (CAT UNIT)
          (NUMERAL ((WORD 90) (CAT NUM))))))
```

5.3. Semantic processing

5.3.1. Description Logic

The semantic processing subsystem uses the Description Logic system Loom (Brill, 1993; MacGregor, 1991) as the basis for all its processing. Loom is used in three different ways:

- to express knowledge about the domain of discourse;
- to build semantic representations of natural language sentences;
- to classify those semantic representations according to a hierarchy of result categories.

Loom implements a subset of first order logic. It provides deductive support for declarative knowledge expressed as concept and relation definitions and instances, through several inference services that are active at all times. Among the most important ones are:

- Concept classification: Loom builds a hierarchy of all concept and relation definitions given, based on logical subsumption, and it classifies any new concept/relation definition with respect to the existing hierarchy.
- Instance classification: Loom uses truth maintenance technologies to dynamically infer conceptual affiliation of instances based on given definitions. It also infers new facts about the instances built, facts that are warranted by the conceptual definitions.
- Consistency check: Loom ensures at all times that the instances built, as well as all concept and relation definitions, are logically consistent.

5.3.2. Upper Model and Geometry knowledge bases

Knowledge about the domain of discourse is represented as Loom definitions and productions. The definitions introduce new concept and relation terms in the semantic model. These terms are used to specify semantic constraints that need to be applied on the representations built by the system. The productions perform additional inferences over the constructed semantic representations.

This knowledge is split in two knowledge bases: the Upper Model and the Geometry knowledge base. This modularity makes the development of a knowledge base for a new domain easier, by allowing the developer to reuse the upper model and replace only the domain specific knowledge base.

The upper model knowledge base contains definitions of generic concepts and relations, which that are not directly related to the geometry domain. Its organization is loosely based on the Generalized Upper Model hierarchy developed as part of the Penman Project (Bateman & al, 1994). It builds two separate hierarchies, one for concepts and one for definitions. The concepts in these hierarchies serve as anchors for concepts in the domain specific knowledge base.

As an illustration, a simplified version of some of the concept and relation definitions necessary for building a semantic representation of the previously considered sentence, repeated here:

(52) The measure of a right angle is 90 degrees.

is given in example (53):

```
(53)  (defconcept Number)
      (defconcept Thing)
      (defconcept Spatial
        :is-primitive Thing)
      (defconcept Abstraction
        :is-primitive Thing)
      (defconcept Measure-Value
        :is-primitive Abstraction)
      (defconcept Unit
        :is-primitive Abstraction)

      (defrelation relation)
      (defrelation participant
        :is-primitive relation)
      (defrelation attribute
        :is-primitive participant)
      (defrelation attribuend
        :is-primitive participant)
      (defrelation belongs-to
        :is-primitive relation)
      (defrelation measure-of
        :is-primitive belongs-to)
      (defrelation measure
        :is (:inverse measure-of))

      (defconcept configuration
        :is-primitive (:and Thing
                            (:all participant Thing)))
      (defconcept Being&Having
        :is-primitive (:and Configuration
                            (:at-most 2 participant)))
      (defconcept Ascription
        :is (:and Being&Having
                  (:exactly 1 attribuend)
                  (:exactly 1 attribute)))

      (implies Ascription
               (:same-as attribute attribuend))
```

Concepts and relations specific to the domain of discourse, in our case geometry, are part of the geometry knowledge base. They may be defined independently of the upper model, or may be defined as subconcepts of elements of the upper model, as needed.

Taking again the sentence in example (52) the needed geometry-specific definitions are:

```
(54)  (defconcept Geometry-Unit
        :is (:and Unit
                  (:one-of 'degree 'meter 'centimeter)))
      (defconcept Angle-Unit
        :is (:and Geometry-Unit
                  (:one-of 'degree 'radian)))

      (defrelation value
        :range Number)
      (defrelation unit
        :range Geometry-Unit)

      (defconcept Geometry-Measure
        :is (:and Measure-Value
                  (:exactly 1 value)
                  (:the unit Geometry-Unit)))
      (defconcept Angle-Measure
        :is (:and Geometry-Measure
                  (:the unit Angle-Unit)))

      (defconcept Geometry-Object
        :is-primitive (:and Spatial
                            (:all measure Geometry-Measure)))
      (defconcept Right
        :is-primitive Geometry-Object)
      (defconcept Angle
        :is-primitive (:and Geometry-Object
                            (:the measure Angle-Measure)))
      (defconcept Right-Angle
        :is (:and Right Angle))
```

The two concept hierarchies are depicted more intuitively in Figure 5-3. The dashed arrows indicate subsumption relations.

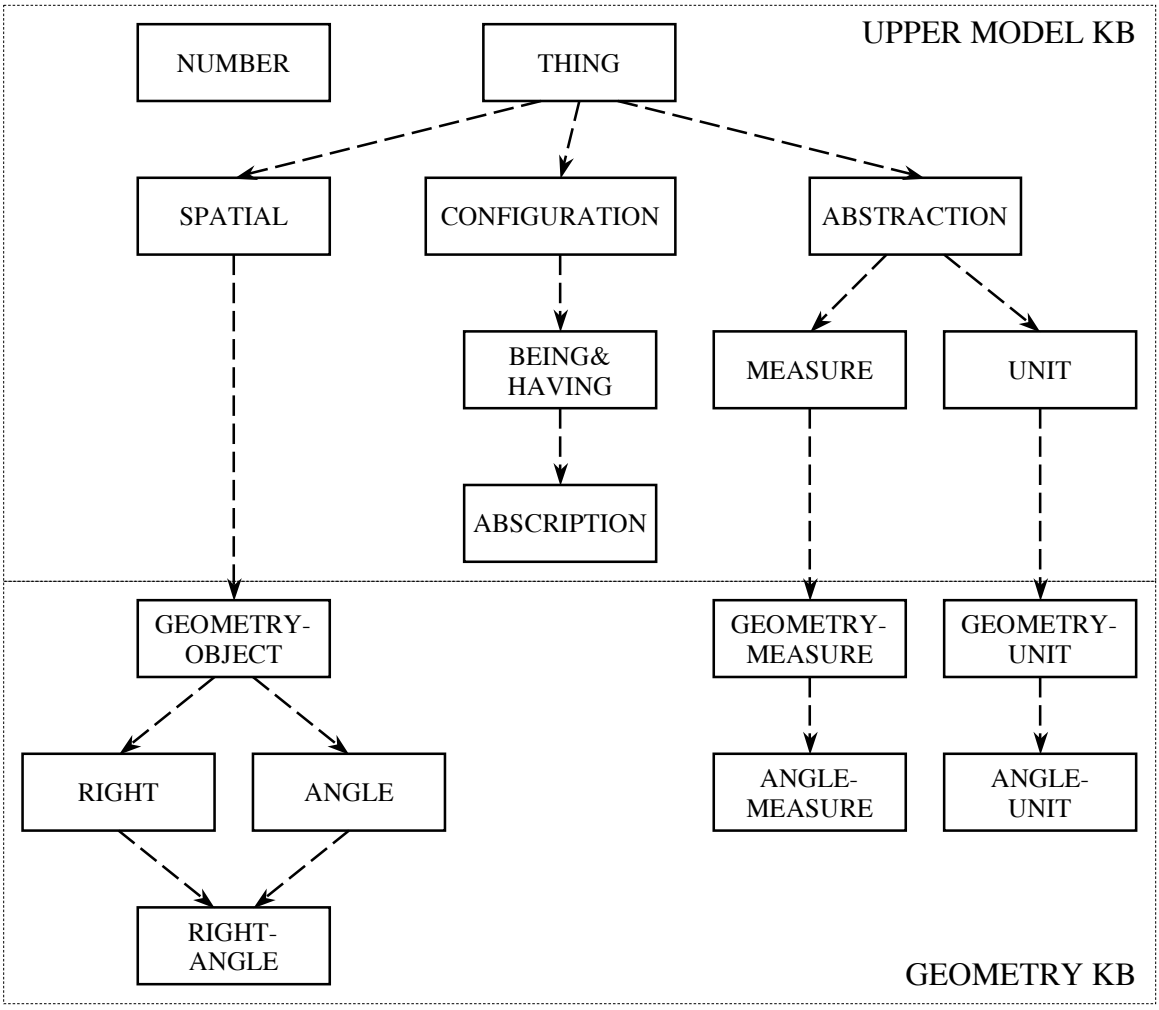

Figure 5-3. Example of partial upper model and geometry concept hierarchy

5.3.3. Semantic representation

The usual way to represent natural language meaning in a Description Logic system is to use formulas allowed by the logic language. Those formulas assert predicates over variables that stand for discourse referents. The predicates stand for concepts and relations present in the sentence and defined in the knowledge base.

Under such an approach, the compositional process of building the semantics for a sentence consists in combining formulas for smaller constituents into larger formulas. The combination process is non-linear, and thus needs a mechanism to specify which elements of the formulas need to be combined together. Such a mechanism is usually provided by lambda calculus.

An alternative approach, which is taken in our system, is to use Loom instances to represent discourse referents and play the role of variables and constants in First Order Logic language. Then concepts and relations are then asserted directly on these instances. This way natural language semantics is represented using Loom's model language.

This approach presents several advantages. First, since these instances are concrete Loom objects, predicates can be asserted directly in Loom, starting with the simplest language structures. Then the process of compositional building of the semantic representation of larger structures out of smaller ones is greatly simplified. Instead of having to keep formulas that need to be combined together, all that is needed is to keep the Loom instances associated with the phrases they represent the meaning of. Combining two such instances to form a new instance for a larger phrase consists of either connecting them through a new relation or combining them into a single instance. These operations can also be performed directly in Loom, and thus the result is always a single instance with new properties.

Asserting predicates directly into the Description Logic system has an additional benefit. Since Loom's inferential services are available in a forward-chaining mode, they will be applied on each instance after each new assertion. Thus the new instance will first be checked for logical consistency. Then it will be reclassified. The new classification can trigger other inference rules or production rules (used for cases that cannot be easily expressed in the definitional language). As a result of applying the new rules, new properties might be asserted on the instance, which can restart the process, in a forward-chaining inference process. This mechanism has the potential to model naturally the explicit reasoning process that people perform when understanding natural language.

Using this mechanism also means that semantic constraints are applied early in the parsing process. These constraints eliminate many of the parse branches that the syntactic grammar would allow, thus pruning the search space, with potential computational benefits.

Finally, the fact that, unlike a logic formula, the semantic representation expressed as a collection of Loom instances lacks a linear structure provides yet another benefit. Since the same basic meaning can be expressed in a variety of different ways even at the logic level, it becomes much easier to recognize whether a desired semantic content is present in the given sentence when the representation can be examined quickly from various angles.

5.3.4. Example of semantic structure

If we take again the sentence in example (52) repeated below, and we use concepts from examples (53) and (54), the desired semantic representation description using Loom's model language is given in example (56).

(55) The measure of a right angle is 90 degrees.

```
(56)  (tell (:about measure-1
              (:create Angle-Measure)
              (unit 'degree)
              (value 90)
              (measure-of angle-1)))
      (tell (:about angle-1
              (:create Right-Angle)
              (measure measure-1)))
      (tell (:about being&having-1
              (:create Ascription)
              (attribute measure-1)
              (attribuend measure-1)))
```

The same semantic representation is shown using a more intuitive diagram representation in Figure 5-4. Concepts are represented with rectangles, while instances are represented with ovals. The labeled links represent relations between instances. The dashed links represent taxonomic classifications of instances with respect to the concept hierarchy.

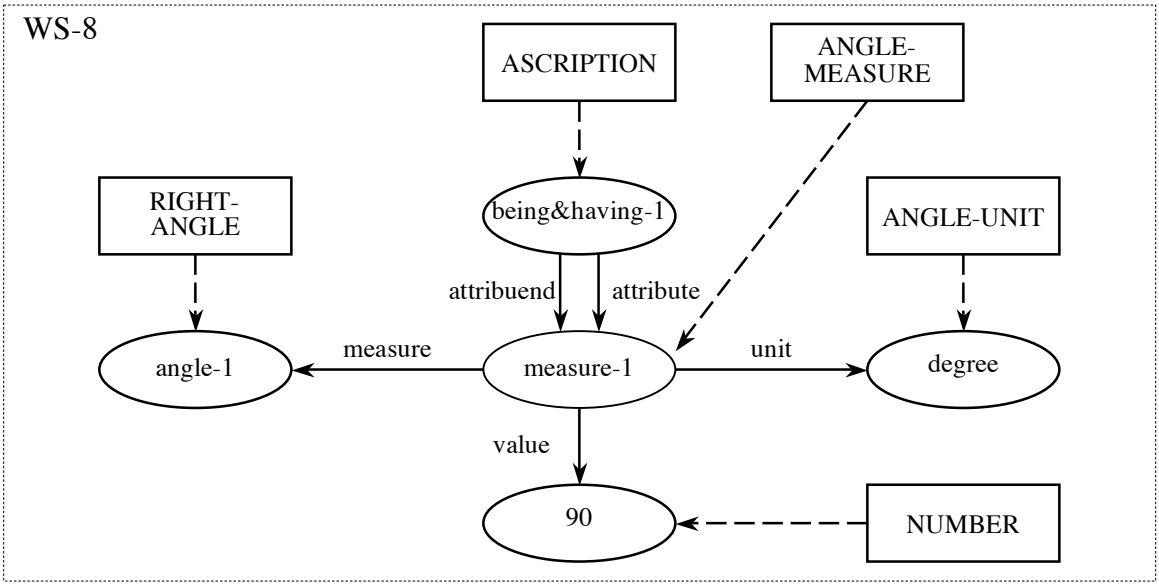

Figure 5-4. Semantic representation for sentence 'The measure of a right angle is 90 degrees.'

One important aspect of our approach, mentioned in section 5.3.3 and illustrated by this example, is that semantic referents that would be represented through variables in First Order Logic are instead represented through Loom instances. For instance the expression `'a right angle'` is represented by instance `angle-1` instead of a universally quantified variable. One drawback of this approach is that there is no way to attach quantifiers to these instances, thus potentially limiting the expressiveness of the language.

Quantifiers could however be explicitly expressed through special-purpose predicates representing the quantifier concepts. For instance one could have a unary `'all'` predicate that can be asserted on instance `angle-1` in the previous example. This could differentiate it from a sentence like:

(57) The measure of the right angle is 90 degrees.

where the similar instance would rather have a 'the' predicate instead.

Like for instance it will not lead, once it was asserted, to the logic system inferring that all right angles have a measure of 90 degrees. This is not a problem in our application because we do not expect the semantic representations to have any logical effect on the knowledge base, as they would have in a knowledge acquisition application for instance. For the purpose of recognizing semantic content of students' input in a tutoring application the lack of logical effect is actually desirable, since students' misconceptions could lead to the introduction of inconsistencies in the knowledge base.

5.3.5. Linguistic inference

The linguistic inference module creates the interface between the feature structure unifier and the Description Logic module. In the process, it also performs two additional inference processes that rely on a combination of linguistic context and domain knowledge: reference resolution and metonymy resolution, which are presented in section 5.3.8 and 5.3.9 respectively.

The interaction between syntax and semantics is mediated by semantic restriction statements attached to rules in the unification grammar. These statements ensure that the right semantic representation is built compositionally from representations for right-hand side components, and that a reference to the built representation is kept in the feature structure. The statements are interpreted by the feature structure unifier as special constructs, which require Lisp function calls.

At each step in the parsing process the semantic representation for a new phrase (the left-hand side non-terminal) is generated through one of four different methods:

- It is newly created by calling function `create-semantics`, in case the component is one of the lexical elements. A special case is when it is created as a new measure, out of components that are a numeric value and possibly a measure unit.
- It is created through the combination (or merging) of the representations of two components through a call to `combine-semantics`.
- It is created by connecting two components through a semantic relation, through a call to `connect-semantics`, when one component is a functional role of the other.
- It is combined in a sequence of representations through a call to `collect-semantics`, when the component representations are not directly logically connected, like in some multi-clause sentences.

Specific information about concepts and relations to be used in the semantic interpretation process are derived from lexical entries for the corresponding words.

To illustrate the process, let's take again the grammar in example (45) and (46) and augment it with semantic restriction statements. The resulting grammar is shown in examples (58) and (59) below:

```
(58)  (<S> ==> (<NP> <VP>)
           ((x0 = x2)
            ((x0 subject) = x1))
            ((x0 semantics) <= (connect-semantics
                                 (x2 semantics)
                                 (x2 subject sem-role)
                                 (x1 semantics))))
      (<NP> ==> (<Det> <N1>)
            ((x0 = x2)
             ((x0 determiner) = x1)))
      (<NP> ==> (<N1>)
            ((x0 = x2)))
      (<N1> ==> (<Adj> <N1>)
            ((x0 = x2)
             ((x0 attribute) = x1))
             ((x0 semantics) <= (combine-semantics
                                  (x2 semantics)
                                  (x1 semantics))))
      (<N1> ==> (<Num> <N1>)
            ((x0 = x2)
             ((x0 numeral) = x1))
             ((x0 semantics) <= (create-measure
                                  (x1 semantics)
                                  (x2 semantics))))
      (<N1> ==> (<N1> <PP>)
            ((x0 = x1)
             ((x0 modifier) = x2))
             ((x0 semantics) <= (combine-semantics
                                  (x2 semantics)
                                  (x1 semantics))))
      (<N1> ==> (<N>)
            ((x0 = x1)))
      (<PP> ==> (<P> <NP>)
            ((x0 = x1)
             ((x0 object) = x2))
             ((x0 semantics) <= (connect-semantics
                                  (x1 semantics)
                                  (x1 relation)
                                  (x2 semantics))))
      (<VP> ==> (<V> <NP>)
            ((x0 = x1)
             ((x0 object) = x2))
             ((x0 semantics) <= (connect-semantics
                                  (x1 semantics)
                                  (x1 object sem-role)
                                  (x2 semantics))))
```

```
(59)  (<Det> ==> (%)
              (((x1 cat) = det)
               (x0 = x1)))
      (<N> ==> (%)
           (((x1 cat) = n)
            (x0 = x1))
            ((x0 semantics) <= (create-semantics
                                  (x1 concept)))))
      (<P> ==> (%)
           (((x1 cat) = p)
            (x0 = x1))
            ((x0 semantics) <= (create-semantics
                                  (x1 relation)))))
      (<Adj> ==> (%)
              (((x1 cat) = adj)
               (x0 = x1))
               ((x0 semantics) <= (create-semantics
                                    (x1 concept)))))
      (<V> ==> (%)
           (((x1 cat) = v)
            (x0 = x1))
            ((x0 semantics) <= (create-semantics
                                  (x1 concept)))))
      (<Num> ==> (%)
              (((x1 cat) = num)
               (x0 = x1))
               (x0 semantics) <= (create-semantics
                                   (x1 concept)
                                   (x1 value)))))
      (<N> ==> (%)
           (((x1 cat) = unit)
            (x0 = x1))
            (x0 semantics) <= (create-semantics
                                (x1 concept)
                                (x1 value)))))
```

This example grammar makes a number of simplifications, like the fact that determiners do not add semantic information, or that all number-noun combinations are measures, or that adjectives do not combine with nouns in functional roles. The grammar relies on additional information provided in the lexicon about the concepts and relations involved, both for the respective lexical entries, and for various functional arguments they might have:

```
(60)  ((:word "the")
       (:cat det))
      ((:word "measure")
       (:cat n)
       (:concept Measure-Value))
      ((:word "of")
       (:cat p)
       (:relation belongs-to))
      ((:word "a")
       (:cat det))
      ((:word "right")
       (:cat adj)
       (:concept Right))
      ((:word "angle")
       (:cat n)
       (:concept Angle))
      ((:word "is")
       (:cat v)
       (:concept Being&Having)
       (:arguments ((subject object))))
       (:semantic-roles ((attribuend attribute))))
      ((:word "90")
       (:cat num)
       (:value 90)
       (:concept Number))
      ((:word "degrees")
       (:cat unit))
       (:value degree)
       (:concept Unit))
```

5.3.6. Semantic contexts

Using the Description Logic's model language to represent natural language semantics during the parsing process gives rise to an additional complication. The parser is a parallel parser, where various tentative phrase structures are built in parallel and kept in the chart for further validation and/or combination with other structures. This process leads to a situation where several different semantic representations are also built in parallel, one for each arc in the chart. All these parallel representations need to be kept separate from one another.

The system accomplishes this separation using the mechanism of workspaces provided by Loom. Workspaces are partitions over the space of instance level assertions. Thus assertions made within a given workspace are seen only in that workspace. Additionally, workspaces come with an inheritance mechanism. When a new workspace is created it can inherit from other existing workspaces. In such a case all assertions in the original workspaces will also be visible in the new workspace.

Our approach uses workspaces to create separate semantic contexts for each new semantic representation structure the logic system builds. The system also uses the inheritance mechanism among workspaces to efficiently combine existing semantic contexts into new contexts, when two phrases are put together to form a larger structure.

The specific way in which semantic contexts are used and combined can be seen when examining a specific example in section 5.3.7.

5.3.7. Example of compositional build

In order to see the compositional semantics process at work, we'll take again the last step in the parsing of sentence in example (52), repeated here:

(61) The measure of a right angle is 90 degrees.

Based on the grammar rules and the lexicon above, the parser recognizes the subject 'The measure of a right angle' and the verb phrase 'is 90 degrees', and creates the feature structures in example (49) and (50), completed with semantic information:

```
(62) ((WORD "measure") (CAT N) (CONCEPT MEASURE-VALUE)
      (SEMANTICS ((CONTEXT WS-4) (CONTENT MEASURE-1)))
      (DETERMINER ((WORD "the") (CAT DET)))
      (MODIFIER
        ((WORD "of") (CAT P) (RELATION BELONGS-TO)
         (SEMANTICS ((CONTEXT WS-3) (CONTENT THING-1)))
         (OBJECT
           ((WORD "angle") (CAT N) (CONCEPT ANGLE)
            (SEMANTICS ((CONTEXT WS-2) (CONTENT ANGLE-1)))
            (DETERMINER ((WORD "a") (CAT DET)))
            (ATTRIBUTE ((WORD "right") (CAT ADJ)
                        (CONCEPT RIGHT)
                        (SEMANTICS ((CONTEXT WS-1)
                                    (CONTENT RIGHT-1)))))))))))

(63) ((WORD "is") (CAT V) (CONCEPT BEING&HAVING)
      (SEMANTICS ((CONTEXT WS-7) (CONTENT BEING&HAVING-1)))
      (SUBJECT ((SEM-ROLE ATTRIBUEND)))
      (OBJECT
        ((WORD "degree") (CAT UNIT) (VALUE DEGREE)
         (CONCEPT UNIT) (SEM-ROLE ATTRIBUTE)
         (SEMANTICS ((CONTEXT WS-6) (CONTENT MEASURE-2)))
         (NUMERAL ((WORD 90) (CAT NUM) (VALUE 90)
                   (CONCEPT NUMBER)
                   (SEMANTICS ((CONTEXT WS-5)
                               (CONTENT 90))))))))
```

The CONTENT fields in the above feature structures point to Loom instances that stand for the discourse referents in the text. The CONTEXT fields give the names of the Loom workspaces where these instances are aggregated. Thus there are two final contexts, WS-4 keeping the structures in example (64) for the subject, and WS-7 with the structures in example (65) for the verb phrase. The final contexts inherit from other contexts that contain part of the two structures.

```
(64)  (tell (:about measure-1
              (:create Angle-Measure)
              (belongs-to angle-1)
              (measure-of angle-1)))
      (tell (:about angle-1
              (:create Right-Angle)
              (measure measure-1)))

(65)  (tell (:about measure-2
              (:create Angle-Measure)
              (unit 'degree)
              (value 90)))
      (tell (:about being&having-1
              (:create Being&Having)
              (attribute measure-2)))
```

The final structures for the two components are illustrated in Figure 5-5 below.

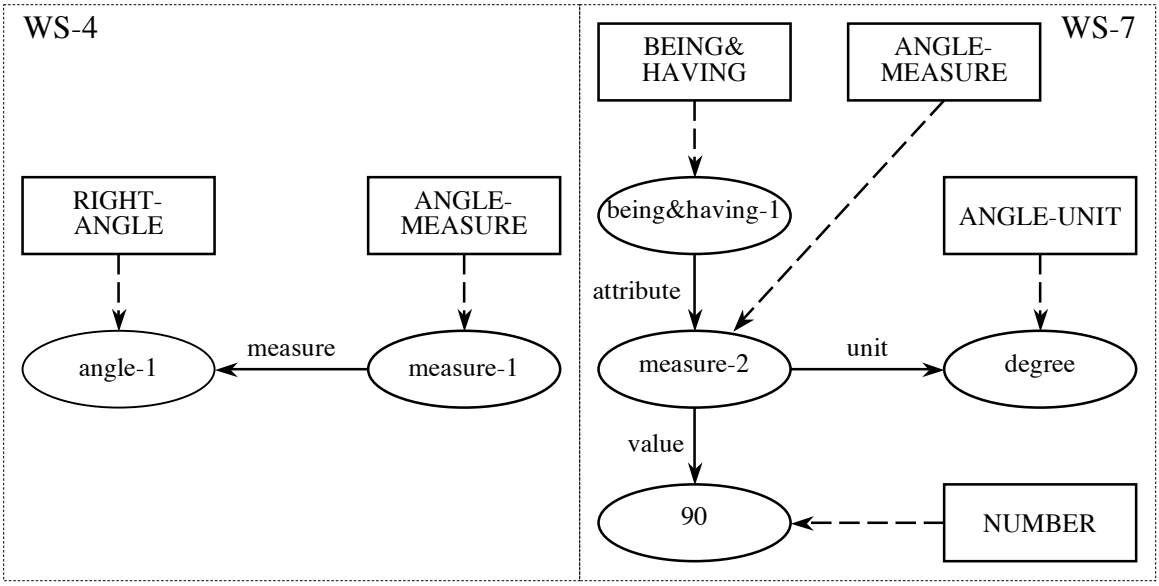

Figure 5-5. Semantic representation for the subject and the verb phrase of sentence 'The measure of a right angle is 90 degrees.'

Having these structures, the parser applies the grammar rule for <S> repeated in example (66):

```
(66)  (<S> ==> (<NP> <VP>)
          ((x0 = x2)
           ((x0 subject) = x1)
           ((x0 semantics) <= (connect-semantics
                               (x2 semantics)
                               (x2 subject sem-role)
                               (x1 semantics)))))
```

The `connect-semantics` function creates a new context `WS-8` that inherits all assertions from contexts `WS-4` and `WS-7`. In this new context the function instructs the logic system to connect instances `being&having-1` and `measure-1` through a relation given by the semantic role of the verb's subject, in this case `attribuend`. In the process,

the logic system checks whether instance measure-1 is logically compatible with instance being&having-1 as filler for relation attribuend. In this case it is.

```
(67)  (tell (:about being&having-1
              (:create Being&Having)
              (attribute measure-2))
              (attribuend measure-1)))
```

The resulting structure, shown in example (67), is classified by Loom as belonging to the more specific concept Ascription, and as a result the implication rule for Ascription is triggered. This rule will try to combine the two measure instances, measure-1 and measure-2, into a single instance.

Again the logic system will ensure that the resulting entity is not logically incoherent. Thus the system is able to obtain the target semantic representation shown before in Figure 5-4.

5.3.8. Reference resolution

Reference resolution is the mechanism that solves linguistic references. Linguistic reference is the phenomenon where certain noun phrases are used to denote elements of the domain of discourse, either explicitly realized in natural language discourse, or implicitly present in the discourse context.

The solution implemented in our approach deals only with references to elements explicitly denoted in previous discourse, called antecedents. However due to its semantic nature, it can be easily extended to deal with implicit elements, once a model of the discourse context is implemented in Loom's model language.

References can come in a variety of linguistic forms, some of which are shown in the examples below:

```
(68)  a) It's a triangle, so its angles sum to 180.

      b) These angles' measures add to 180 because they form a
         linear pair.

      c) The sum of adjacent angles is 180 degrees if those angles
         form a line.

      d) The sum of the measures of two adjacent angles is equal
         to the measure of the angle formed by the two angles.

      e) If adjacent angles form an angle, its measure is their
         sum.
```

For example in sentence a) above the possessive pronoun 'its' refers to antecedent 'a triangle'. In example b) the personal pronoun 'they' refer to 'these angles'. Sentence c) uses a referential noun phrase with a demonstrative article 'those angles' to denote the same element as the antecedent 'adjacent angles'. Similarly in sentence d) a noun phrase with a definite determiner 'the two angles' is used to identify the same discourse element as the antecedent 'two adjacent angles'. Example e) shows a case where two possessives are used in the same clause to refer to

two different antecedents, the correct correspondence being made based on number constraints.

Unlike many syntactic based systems, the resolution of referents to antecedents is done in our approach at the semantic level. That is, rather than trying to connect the linguistic constructs of the elements involved through some kind of "binding" connection, we simply try to combine or merge the semantic representation of the referent with that of the antecedent. This mechanism has the advantage that when the system attempts to combine two discourse referents, it will make sure that all semantic constraints associated with the two discourse referents are enforced. Thus, elements that are incompatible will fail the merge. This use of semantics takes care both of number restrictions, as well as all other semantic features, like taxonomic compatibility between the concepts involved.

To see how this works, if we take sentence (68) a), we first create the semantic representation for the two clauses, as seen in Figure 5-6. Some elements of these representations are the result of additional processes. For instance `measure-1` is the result of the metonymy resolution process (see section 5.3.9), while instance `degree` is the result of an inference process based on the fact that the measure is connected to an angle.

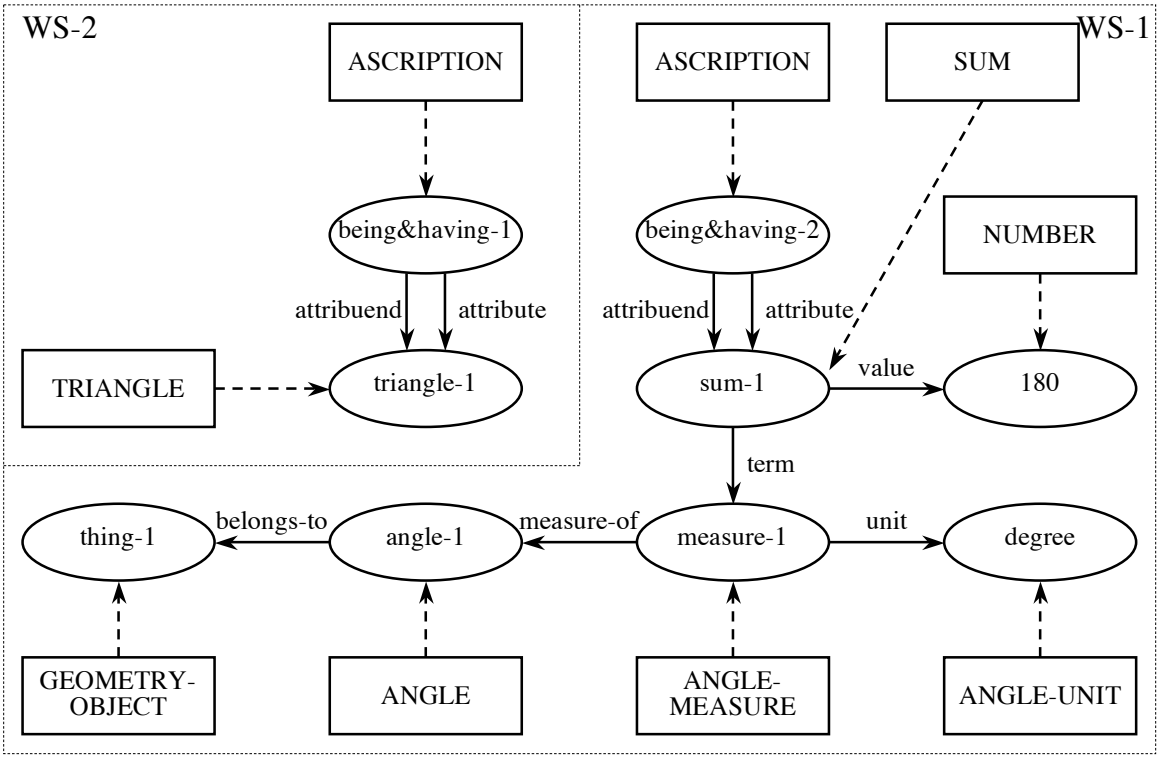

Figure 5-6. Semantic representation for 'It's a triangle' and 'so its angles sum to 180'.

Then, when combining the two clauses into a single representation, the reference resolution mechanism will try to combine `thing-1` (the representation of the possessor part of '`its`') with `triangle-1.` The two instances are semantically compatible (concept `Triangle` is a subconcept of `Thing`), so the merge succeeds, resulting in the structure shown in Figure 5-7.

This mechanism allows for the correct choice of referent-antecedent pairs in example (68) e), just based on number constraints. The semantic representation of '`its`' is marked as a `Single` concept, and thus will fail to merge to the discourse referent of '`adjacent angles`', while the semantic representation of '`their`' is marked for `Collective`, and will fail to merge with '`an angle`'.

While this case can also be solved in a syntactic approach by just checking the `Number` feature of the elements involved, example (68) b) is more difficult to deal with. In our approach, because the semantic representation of '`they`' has a `part` role in the `Form` configuration of a whole that's a `Linear-Pair`, the logic system can infer it has to belong to the `Angle` concept. Then when it tries to combine it to the two antecedent candidates '`measures`' and '`angles`', only the second one will succeed.

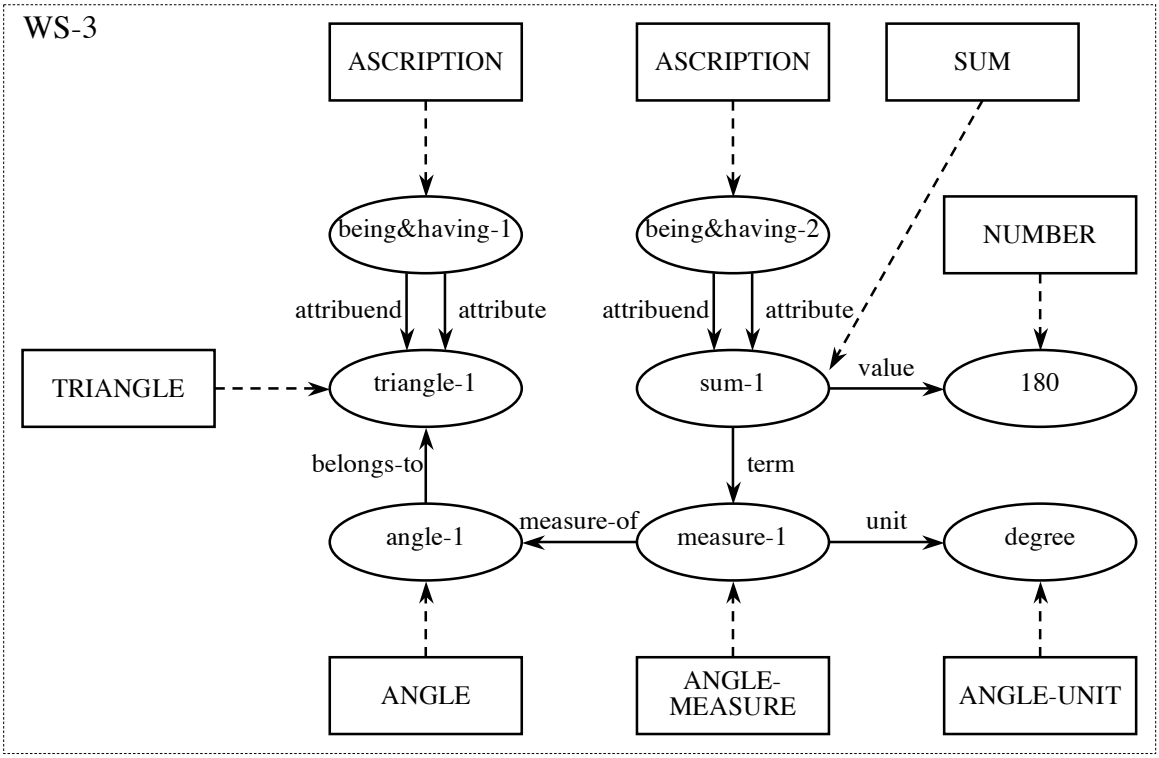

Figure 5-7. Semantic representation for 'It's a triangle, so its angles sum to 180.'

There are however a number of syntactic restrictions that have to be observed when choosing which pairs of "referent-antecedent" to try to combine. Those constraints are expressed in the various binding theories associated with main linguistic theories. Our approach gives an implementation of the binding theory from HPSG (Pollard & Sag, 1994), by using lists of discourse referents associated with the feature structures of various sentence elements.

There are three different kinds of referents that observe different syntactic restrictions: anaphors (reflexives and reciprocals), personal pronouns, and non-pronominals. In a simplified form, the binding theory says that anaphors have to be bound within a local domain, personal pronouns have to be free within a local domain, and non-pronominals have to be totally free in any argument structure.

In order to implement the restrictions, our approach keeps lists of the candidate referents for each syntactic constituent, together with a list of potential antecedents. As the parsing proceeds, various grammar rules try to solve some of the candidate referents against some of the potential antecedents, and keep track of the remaining candidates.

5.3.9. Metonymy resolution

Metonymy is the phenomenon of using a concept to stand for a different one to which the first one is related in some implied way. In our domain we have a limited number of such cases, the most widespread being two cases: using angles instead of their measures, and using sets instead of their elements. Both of them appear in example (35) a) repeated below:

(69) `The sum of a linear pair of angles is 180.`

Here the '`sum`' cannot be of a '`linear pair`' itself, but instead it's the sum of the measures of the angles of the linear pair. So two intermediary concepts are missing, '`the measures`' and '`the angles`'.

Again, as in the case of reference resolution, our approach deals with metonymies at the semantic level. The very fact of recognizing we have a metonymy relies on being able to identify there is a mismatch at the conceptual level between elements connected syntactically. Our solution deals with the most frequent cases that appear in our corpus. For those cases, the implemented solution intervenes at the level of `combine-semantics` operation presented in section 5.3.5. It identifies those cases by looking for certain combinations of concepts involved. And if such a combination is found, the mechanism first looks for the missing discourse referents in the immediate vicinity of the semantic representations involved. If nothing is found, it explicitly supplies the missing referents.

Thus, in the example above, the mechanism will first look to see if the set denoted by '`pair`' has any elements, and it finds a discourse referent for '`linear`'. It then uses it and looks for a '`measure`' referent associated with it. Since none is found, it is supplied and connected correctly to both the referent for '`linear`' and that for '`the sum`'.

5.3.10. Classification

The semantic representation built by the system for a given input sentence needs to be interpreted by the tutor according to a given set of response classes, so that it can give appropriate feedback to the student. Since these classes also deal mainly with the concepts involved in the sentence, it is natural to do the process in the same framework as the semantic interpretation.

The solution adopted takes advantage of Loom's built-in classification mechanism. Thus, the set of classes that need to be recognized is represented as a taxonomy of Loom concept definitions. Then, given this taxonomy and a representation of the sentence's semantics as an instance configuration in Loom's model language, the logic system will automatically classify it according to the given taxonomy.

For an example of how the classification process works we can take again the sentence in example (68) a). Examples of partial expressions of the full reason taken from our corpus are given in example (70).

(70) a) `The measure of these angles is 180.`

b) `The sum of the measures is 180 degrees.`

c) `The angles of a triangle are 180 degrees.`

d) `The angles add up to 180.`

e) `The angles of a triangle sum to 180 degrees.`

A partial taxonomy of classes needed for recognizing various incomplete forms of presenting the `TRIANGLE-SUM` and other theorems are shown in Figure 5-8. The concept boxes are accompanied by examples of sentences that express them. These classes are a small part of a hierarchy of about 220 classes currently present in our application. The partial classes represented approximately ¾ of the classes that the NLU system generated in a recent evaluation test (see Chapter 7).

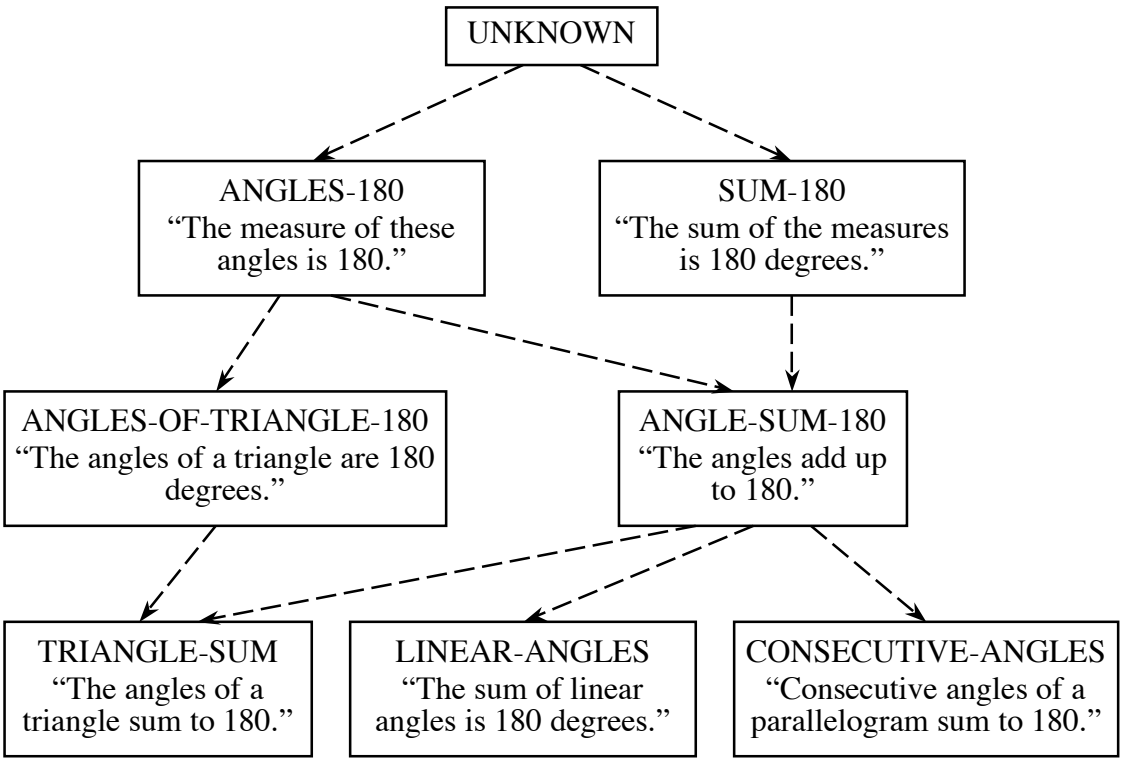

Figure 5-8. Example of partial class taxonomy for the 'Triangle Sum' theorem with associated sentence examples

The classes in the taxonomy give more and more refined concept definitions for what constitutes a valid semantic representation for each respective class. They could look like:

```
(71)  (defconcept Angles-180-Reason
        :is (:and Ascription
                  (:the attribuend
                        (:and Angle-Measure
                              (:the measure-of Angle)))
                  (:the attribute (:the value 180))))

      (defconcept Angles-of-Triangle-180-Reason
        :is (:and Ascription
                  (:the attribuend
                        (:and Angle-Measure
                           (:the measure-of
                              (:and Angle
                                 (:some vertex-of Triangle)))))
                  (:the attribute (:the value 180))))

      (defconcept Triangle-Sum-Reason
        :is (:and Ascription
                  (:the attribuend
                     (:and Sum
                        (:some term
                           (:and Angle-Measure
                              (:the measure-of
                                 (:and Angle
                                    (:some belongs-to Triangle)))))))
                  (:the attribute (:the value 180))))
```

The semantic representation of sentence (68) a), shown in Figure 5-7, can be translated into Loom's model language as:

```
(72)  (tell (:about triangle-1
              (:create Triangle)))
      (tell (:about being&having-1
              (:create Being&Having)
              (attribuend triangle-1)
              (attribute triangle-1)))
      (tell (:about angle-1
              (:create Angle)
              (belongs-to triangle-1))
      (tell (:about measure-1
              (:create Angle-Measure)
              (measure-of angle-1)))
      (tell (:about sum-1
              (:create Sum)
              (term measure-1)
              (value 180)))
      (tell (:about being&having-2
              (:create Being&Having)
              (attribuend sum-1)
              (attribute sum-1)))
```

It can be easily checked that instance `being&having-2` above has all the elements required in the definition of `Triangle-Sum-Reason`, so Loom will classify it as such.

Chapter 6 Advantages of Logic-Based Approach over Alternatives

This chapter will discuss some of the advantages of using our approach of unification-based syntax with logic-based semantics over alternative approaches when dealing with some of the problems presented previously.

6.1. Natural, consistent modeling

One property that is often overlooked when evaluating an NLU system is naturalness of modeling natural language and the domain of discourse within the proposed framework. Naturalness might not seem important when looking at small demonstration systems. But when the task is to develop a real-life system that has to be maintained and improved over a long period of time, naturalness of the model becomes an essential feature in the development process. When having to deal with fixing a problem or extending the system to cover new cases, a natural approach will allow the developer to quickly understand why the system fails and how to make the necessary change. In our case this naturalness is judged by similarity to how humans model natural language and the domain of discourse in their minds.

Syntactic-based approaches also capture the main language structures in a way similar to how humans think about them. They usually consist of a grammar that models the syntactic patterns of the language, and a lexicon that describes the basic atoms of the language, the words. Thus, when a problem of syntactic and/or lexical nature occurs, it is usually not very hard to identify it and bring appropriate changes to the system. However, parsing a sentence under a syntax-only framework many times leads to a considerable number of syntactically valid structures, which have to be rejected on semantic grounds. In absence of a natural way to express semantic constraints, the only solution for the developer is to alter the grammar and/or the lexicon to reject some of those undesired structures. This process blurs the separation between syntax and semantics, and leads to a model full of idiosyncrasies, where it becomes hard to predict which structures will be accepted and which ones will be rejected. Thus, a good part of the naturalness of the model is lost.

Since humans think of domains of reality in terms of taxonomies of concepts, an approach that allows easy expression of such taxonomies have a lead on naturalness over alternatives. Moreover, basing a taxonomy model on a logic system has two main additional benefits. First, it allows the system to perform many of the quick inferences that a human does when understanding natural language, thus making the modeling

process easier. And second, it permanently checks the consistency of the taxonomy. In the process of developing taxonomies of hundreds or thousands of concepts, one is very probable to introduce inconsistencies in concept definitions, which would lead to unpredictable inferential behavior. Keeping the developer true to the meaning of the developed definitions is a huge help in avoiding later headaches.

A further benefit in the development process comes from using Loom instances instead of variables to stand for discourse referents. The use of instances brings the ability to apply semantic constraints early in the NLU process, similar to when a human would apply them. Thus the understanding process looks more natural to the human developer, because many of the structures that a syntax-only approach would validate never show up, since they are rejected early by semantic constraints.

6.2. Advantages in determining the semantic content of sentences

Chapter 3 presented some of the problems a natural understanding system will have to solve in order to determine accurately the semantic content of a given sentence. This section will show how some of these problems can be solved within the approach presented in this dissertation.

6.2.1. Uniqueness of semantic content over various ways of expression

The generic problem presented in section 3.1 was to have reliable means to determine when different ways of expressing the same meaning are semantically equivalent. In a syntactic approach, different surface realization forms usually lead to different parse results (like different feature structures in a unification approach). These results in turn may lead to different semantic representations, whose equivalence remains to be determined.

Rather than getting different semantic structures whose equivalence has to be determined afterwards, the solution adopted in our approach is to combine the power of the syntactic unification formalism with the inference power of the underlying logic system to actually generate the same Loom semantic structure in cases of semantic equivalence. The specific solutions in some of the cases presented in Chapter 3 are discussed below.

6.2.1.1. Variation of syntactic structure

The clear separation of syntax and semantics in our approach allows for a clean treatment of passive constructions through a reduced number of grammar rules, similar to other syntactic approaches. Moreover, the grammar rule can alter the verb's argument structure, so that the semantic structures for variants as those in example (10) repeated below will be the same.

(73) a) `Two intersecting lines form congruent vertical angles.`

b) `Congruent vertical angles are formed by two intersecting lines.`

A simplified rule that extends the example grammar given in (58) and (59) to deal with passive constructs is shown below:

```
(74)  (<V> ==> (<AUX> <V>)
           (((x1 root) = "be")
            ((x2 form) = 'past-participle)
            (x0 = x2)
            ((x0 form) <= 'passive)
            ((x0 oblique root) = "by")
            ((x0 oblique object) <= (x2 subject))
            ((x0 subject) <= (x2 object))))
```

This rule also requires a second rule for dealing with oblique arguments at the verb phrase level, but that is needed anyway for other verbs that take oblique arguments in active form:

```
(75)  (<VP> ==> (<V> <PP>)
           ((x0 = x1)
            ((x0 oblique) = x2))
            ((x0 semantics) <= (connect-semantics
                                  (x1 semantics)
                                  (x1 oblique sem-role)
                                  (x2 semantics)))))
```

The change in syntactic argument structure performed by the first rule above (`subject` to `oblique object` and `object` to `subject`) also has the effect of switching the semantic roles provided by the verb's lexical entry for the two arguments. Thus, the semantic roles will be in the right place when the semantic representation for the verb phrase is built by the second rule, and the resulting semantic representation will be completely invariant to the verb form.

The problem with sentences in example (11) repeated below, is that prepositional phrases attached to different places in a sentence sometimes do not lead to a significant difference in meaning, but other times they do, as seen in examples (12) and (13). The determination in this case has to be made on semantic grounds.

```
(76)  a) In a triangle angles opposite to congruent sides are
         congruent.

      b) Angles in a triangle opposite to congruent sides are
         congruent.

      c) Angles opposite to congruent sides in a triangle are
         congruent.

      d) Angles opposite to congruent sides are congruent in a
         triangle.
```

In our approach the solution for example (76) can come as a pair of concept definitions that specify additional inferences on the concept elements. The first one takes care of sentences b) and c) by detecting the situation when an object (`angle` in our case) is included in some container (`triangle`) and is in a relation of a certain kind (like `opposite-to` in our case) with another object (`side`). In this case it percolates the "container" `belongs-to` relation to the second object. This way all "contained" objects will have the `belongs-to` relation, and the structure will be independent of where the relation occurred in the input language.

```
(77)  (defrelation opposite-to
        :is-primitive (:and relation
                            (:domain Geometry-Object)
                            (:range Geometry-Object))
        :characteristics (:single-valued :symmetric))
      (defconcept Object-in-Object-Opposite
        :is (:and Geometry-Object
                  (:at-least 1 opposite-to)
                  (:some belongs-to Geometry-Object)))

      (implies Object-in-Object-Opposite
               (:relates belongs-to opposite-to belongs-to))
```

The second definition deals with sentences a) and d) by identifying the container relation at the assertion level, and percolating it down to involved objects.

```
(78)  (defconcept Ascription-Location
        :is (:and Ascription
                  (:at-least 1 belongs-to)))

      (implies Ascription-Location
               (:and (:relates belongs-to
                               attribuend belongs-to))
                     (:relates belongs-to
                               attribute belongs-to)))
```

These definitions can be easily generalized, if they need to apply to a larger class of situations, like other relations besides `opposite-to`. Or they can be specialized if for instance they apply only to a certain subclass of geometry objects.

The ability of our approach to have production rules finely tuned to specific concept combinations allows us to have an elegant solution expressed at the right level of generality for example (14), repeated below:

```
(79)  a) The measures of these two angles are equal.

      b) These two angles are equal in measure.

      c) These two angles have equal measures.

      d) These two angles measure the same.
```

The generic solution here is to make all sentences generate the same semantic structure (plus possibly extra elements that can be ignored) as sentence (79) a). Thus for sentence (79) b) we can define a concept/rule pair that will identify cases of '`equal in some measurable quantity`' and will generate a structure with the meaning of '`equal quantity`'. These definitions will also work for '`equal in length`', '`equal in area`', etc.

```
(80)  (defrelation equal-to
        :is-primitive (:and relation
                            (:domain Thing)
                            (:range Thing))
        :characteristics :symmetric)
      (defconcept Equal
        :is (:at-least 1 equal-to))
      (defconcept Equal-in
        :is (:and Equal (:some location Measure-Value)))

      (implies Equal-in
               (:same-as (:compose equal-to equal-to) location))
```

This rule is generic enough to apply to the entire class of specialized measures, like lengths, areas, etc., as long as those are defined as subconcepts of `Measure-Value`.

Sentence (79) c) can be resolved by defining inferences that would generate an `Ascription` configuration similar to that corresponding to sentence (79) a), from the `Generalized-Possession` configuration generated by the verb '`have`', under appropriate semantic restrictions:

```
(81)  (defconcept Generalized-Possession
        :is-primitive Being&Having)
      (defrelation possessor
        :is-primitive participant)
      (defrelation possessed
        :is-primitive participant)
      (defconcept Attribute-Possession
        :is (:and Generalized-Possession
                  (:at-least 1 attribute)))

      (implies (:and possessed
                     (:domain Generalized-Possession)
                     (:range Measure-Value))
               attribute)
      (implies (:and possessor
                     (:domain Attribute-Possession))
               attribuend)
```

The solution for sentence (79) d) can be solved by defining the concept associated with the verb '`to measure`' to be a subconcept of the generic concept `Generalized-Possession` used for the verb '`to have`', and assigning the semantic role for its object to be an `attribute`. Then we can define an inference rule that would perform the additional necessary reasoning steps that are needed to get the semantic structure of sentence (79) c). Thus sentence (79) d) will be interpreted the same way as if it said '`These two angles have the same measure`':

```
(82)  (defconcept Measure-Process
        :is-primitive Generalized-Possession)

      (implies Measure-Process
               (:and (:some possessed Measure-Value)
                     (:same-as attribute possessed)))
```

Then we ca define '`the same`' as a synonym for '`equal`', and we get the same meaning as for sentence (79) c).

Unlike many syntactic-based frameworks, the semantic representation of sentences in our system is completely independent of the specific syntactic structures used in expressing that meaning. This is true not only at the clause level, like in systems that derive a "deep" predicate-argument structure from the syntactic structure, but also at the sentence level, where the distribution of semantic content among various clauses is also ignored. This independence helps in dealing with a large variety of syntactic forms that are semantically equivalent, or where the differences in semantic content can be ignored, like in example (15) repeated here:

(83) a) The sum of the measures of two complementary angles is 90 degrees.

b) If two angles are complementary, then the sum of their measures is 90 degrees.

c) The measures of two angles sum to 90 degrees, because they are complementary angles.

d) Complementary angles are angles whose measures sum to 90 degrees.

These sentences differ mostly in the clause structure used to realize the same content. The variation over the different ways the implication is expressed can be safely ignored for tutoring purposes. Then sentence a) is a single clause, whose semantic content is built straightforwardly into the structure:

```
(84)  (tell (:about being&having-1
              (:create Being&Having)
              (attribute sum-1)
              (attribuend sum-1)))
      (tell (:about sum-1
              (:create Sum)
              (value 90)
              (unit 'degree)
              (term measure-1)))
      (tell (:about measure-1
              (:create Geometry-Measure)
              (count 'collective)
              (measure-of angle-1)))
      (tell (:about angle-1
              (:create Angle)
              Complementary
              (count 'collective)
              (measure measure-1)))
```

Sentences b), c), and d), when analyzed clause by clause, will generate almost the same content but split in two parts in different ways. Thus for sentence b) for instance the split will be:

```
(85)  (tell (:about being&having-1
              (:create Being&Having)
              (attribute angle-1)
              (attribuend angle-1)))
      (tell (:about angle-1
              (:create Angle)
              Complementary
              (count 'collective)))
```

and:

```
(86)  (tell (:about being&having-2
              (:create Being&Having)
              (attribute sum-1)
              (attribuend sum-1)))
      (tell (:about sum-1
              (:create Sum)
              (value 90)
              (unit 'degree)
              (term measure-1)))
      (tell (:about measure-1
              (:create Geometry-Measure)
              (count 'collective)
              (measure-of thing-1)))
      (tell (:about thing-1
              (:create Thing)
              (count 'collective)
              (measure measure-1)))
```

Then the reference resolution mechanism described in section 5.3.8 is applied, resulting in the identification of instance `thing-1` from the second clause with instance `angle-1` from the first clause. As a consequence the same structure as in example (84) is produced, plus one extra instance, `being&having-1`, which can easily be ignored.

6.2.1.2. Variation of content words

Cases like those in example (18) and (19), repeated below, when two words are synonyms only in certain contexts, are solved by giving appropriate definitions to the respective concepts.

```
(87)  a) Angles ABC and BAC are equal.
      b) Angles ABC and BAC are congruent.
(88)  a) The measures of angles ABC and BAC are equal.
      b) *The measures of angles ABC and BAC are congruent.
```

In this case we can define '`congruent`' as a specialized case of '`equal`':

```
(89)  (defrelation equal-to
        :is-primitive (:and relation
                            (:domain Thing)
                            (:range Thing))
        :characteristics :symmetric)
      (defrelation congruent-to
        :is (:and equal-to
                  (:domain Finite-Object)
                  (:range Finite-Object)))
```

Using these definitions, Loom will classify the 'equal' relation in sentence (87) a) as being in fact a 'congruent' relation, and thus will produce the same representation as for sentence (87) b). The same definitions will invalidate sentence (88) b), because the logic system will check whether the entities related by the candidate 'congruent' relation are geometry objects, and in this case they are not.

Moreover, we can add a concept/rule definition that will perform the inference that if the measures of some objects are equal, then the objects themselves are congruent.

```
(90)  (defconcept Equal-Measure
        :is (:and Measure-Value
                  (:at-least 1 equal-to)))

      (implies Equal-Measure
               (:relates congruent-to measure-of
                         (:compose equal-to measure-of)))
```

Using this definition, Loom will be able to create for sentence in example (88) a) the same structure as for sentence (87) b), thus making them semantically equivalent.

Use of generic concepts instead of more specialized ones, like in example (20) repeated below can be dealt with using the same mechanism of taxonomic concept definitions.

```
(91)  a) The line between points A and B measures 10 cm.

      b) The segment between points A and B measures 10 cm.
```

Here, concept 'segment' can be defined as a type of finite line. Further the relation 'measure' can be restricted to apply only to finite objects (since one cannot measure infinite objects, at least in geometry).

```
(92)  (defconcept Line
        :is-primitive Geometry-Object)
      (defconcept Segment
        :is (:and Line Finite-Thing))
      (defrelation measure-of
        :is-primitive (:and belongs-to
                            (:domain Measure-Value)
                            (:range Finite-Thing)))
      (defrelation measure
        :is (:inverse measure-of))
```

Based on these definitions, Loom will infer that the 'line' in example (91) a) also has the property of being 'finite' (since it is the domain of a 'measure' relation), and it

will further classify it as a 'segment', thus producing the same structure as for example (91) b).

Similarly, cases where sentences differ by using an explicit definition instead of the name of the defined concept, like examples (23) to (25) repeated below, can be dealt with by giving the appropriate concept definitions.

(93) a) In a triangle with two congruent sides the base angles are congruent.

b) In an isosceles triangle the base angles are congruent.

(94) a) Adjacent angles on a line sum to 180 degrees.

b) Linear angles sum to 180 degrees.

(95) a) Opposite angles that are formed by two intersecting lines are congruent.

b) Vertical angles are congruent.

Thus, sentences in example (93), (94), and (95) will need definitions of 'isosceles triangles' as being triangles with 2 congruent sides, of 'linear angles' as adjacent angles on a line, and of 'vertical angles' as opposite angles formed by two lines:

(96)
```
(defrelation side
  :is (:and belongs-to
            (:domain Geometry-Object)
            (:range Geometry-Object)))
(defconcept Triangle
  :is (:and Polygon
            (:exactly 3 side)))
(defconcept Isosceles-Triangle
  :is (:and Triangle
            (:at-least 2 side Congruent)))
```

(97)
```
(defrelation adjacent-to
  :is-primitive (:and relation
                      (:domain Geometry-Object)
                      (:range Geometry-Object)))
(defconcept Adjacent-Angle
  :is (:and Angle
            (:at-least 1 adjacent-to)))
(defconcept Angle-on-Line
  :is (:and Angle
            (:some location Line)))
(defconcept Linear-Angle
  :is (:and Adjacent-Angle Angle-on-Line))
```

```
(98)  (defconcept Angle-with-Sides
        :is (:and Angle
                  (:some side Line)))
      (defrelation vertical-to
        :is (:and opposite-to
                  (:domain Angle-with-Sides))
        :implies (:range Angle))
      (defconcept Vertical
        :is (:and Angle
                  (:at-least 1 vertical-to)))
```

6.2.2. Structural ambiguity

Our approach provides ways to deal with a fair number of cases of structural ambiguities, where an approach based solely on syntax would end up with lots of different parses. Thus, if we take example (33) repeated below:

```
(99)  The sum of the measures of the three interior angles in a
      triangle is equal to 180 degrees.
```

As presented in section 3.2.2, the problem here is that a syntactic grammar allows for 6 different valid parses, with no way to choose the one that makes sense semantically. By adding appropriate restrictions to the definitions of the concepts involved, our approach can make some of these combinations invalid during the parsing process. The remaining result will be the valid parse. In our case, we write the semantic representation to restrict sums to only apply to elements that are measurable, and thus eliminate the attachment of prepositional phrase '`of the three interior angles`' to '`the sum`'. We also restrict the containment relation to have geometry objects on both sides, and thus eliminate the attachment of '`in a triangle`' to either '`the sum`' or '`the measures`'.

```
(100) (defconcept Sum
        :is-primitive (:and Measure-Value
                            (:at-least 1 term)
                            (:all term Measure-Value)))

(101) (defconcept Object-in-Location
        :is (:and Object
                  (:some location Geometry-Object))
        :implies Geometry-Object)
```

6.2.3. Reference resolution disambiguation

The semantic restrictions provided by the logic model of the domain of discourse also help with choosing the right antecedent for reference resolution. One such example was presented in section 5.3.8. A more complicated case was given in example (36), repeated below:

```
(102) If the lengths of two sides of a triangle are equal, then
      the measures of the angles opposite them will also be equal.
```

Here there are four possible candidates as antecedent for the pronoun '`them`': '`the lengths`', '`two sides`', '`a triangle`', and '`the measures`'. Constraints of the

Binding Theory implemented in our approach eliminate ‘the measures’, since ‘them’ is a personal pronoun that has to be free within its local domain. Constraints on number eliminate ‘a triangle’, as being singular, while ‘them’ is plural. Then semantic constraints attached to the definition of relation ‘opposite’ can eliminate ‘the lengths’, by restricting geometry objects to only be opposite other geometry objects:

```
(103) (defconcept Object-Opposite
        :is (:and Geometry-Object
                  (:some opposite-to Thing))
        :implies (:all opposite-to Geometry-Object))
```

Chapter 7 Evaluation of NLU performance

In order to assess whether the system was developed to a reasonable degree, we conducted an evaluation of the system's performance. We focused on the accuracy of the system's ability to classify student explanations with respect to its set of explanation categories. The classification task is inherently ambiguous. For some explanations it is difficult even for human raters to determine what the correct categories are. Therefore, rather than compare the labels assigned by the system against a "correct" set of labels, we examined to what extent the agreement between system and human raters approaches that between human raters. This design was used also in earlier studies, for example a study to evaluate an automated method for essay grading (Foltz & al, 1999).

7.1. Evaluation method

To compute the inter-rater agreement, we used the κ (kappa) statistic (Cohen, 1960). The κ statistic provides a measure of how much agreement there is between raters beyond the agreement expected to occur by chance alone. It is defined by the formula:

$$(104)\quad \kappa = \frac{p_0 - p_c}{1 - p_c} = 1 - \frac{1 - p_0}{1 - p_c} = 1 - \frac{q_0}{q_c}$$

Where: p_0 = the proportion of units in which the raters agreed
p_c = the proportion of units for which agreement is expected by chance.
q_0 = the proportion of units in which the raters disagreed
q_c = the proportion of units for which disagreement is expected by chance.

The original κ measure assumes each example in the test set is assigned a single category label. However, in our system it is often the case that a sentence can be labeled with a set of categories that complement each other to cover the entire sentence meaning. Thus we need a way to measure how much two such sets of labels agree. Therefore we used "weighted κ" (Cohen, 1968), a version of the κ statistic that takes into account the degree of difference between labels (or in our case, label sets). Under such a measure a weight is assigned to each pair of sets of labels in the *n* x *n* matrix of joint label assignment frequencies for each pair of raters.

In case of weighted κ, the proportions are computed based on the agreement matrix as:

$$(105)\quad q_0 = \frac{\sum v_{ij} p_{0ij}}{v_{\max}} \qquad q_c = \frac{\sum v_{ij} p_{cij}}{v_{\max}}$$

Where: v_{ij} = the disagreement weight of cell *ij* in the *n* x *n* table
v_{max} = maximum disagreement weight (taken as 1 in our case)
p_{0ij} = proportion of joint labels observed in cell *ij*
p_{cij} = proportion of joint labels in cell *ij* expected by chance

We used three different measures for disagreement between two different sets of labels given by two different raters to the same sentence. First we used a "set equality" measure. Under this measure two sets of labels were considered to agree only if they are identical. That means $v_{ij} = 0$ if the two sets are identical and $v_{ij} = 1$ if they differ in any label.

However, this measure puts too high a penalty on small differences between two sets of labels. Then, as the second measure we used an "overlap" measure, where the degree of disagreement between two sets of labels was computed as the ratio of the number of unshared labels versus the total number of labels, using the formula:

$$v_{ij} = \frac{1}{2}\left(\frac{diff(S_i,S_j)}{card(S_i)} + \frac{diff(S_j,S_i)}{card(S_j)} \right) \quad (106)$$

Where: S_i and S_j are the two sets of labels
$diff(S_i,S_j)$ = the number of labels of S_i which is not shared with S_j
$card(S_i)$ = the total number of labels of S_i

While this measure is able to take into account the degree to which two sets have labels in common, it still has a problem. It considers all labels to be disjoint and equally different from one another. However in our case this is not true. Since we have a taxonomy of categories with various conceptual content, some pairs of categories are more similar than others. And most of them are not totally disjoint, they have part of the meaning content in common. So, as a third measure we used a "weighted overlap" measure that takes into account the semantic dissimilarity between individual labels. In this case the weights become:

$$v_{ij} = \frac{1}{2}\left(\frac{\sum_{S_i} dist(l_k,S_j)}{card(S_i)} + \frac{\sum_{S_j} dist(l_k,S_i)}{card(S_j)} \right) \quad (107)$$

Where: S_i and S_j are the two sets of labels
l_k = each label of each set respectively
$dist(l_k,S_i)$ = the minimum distance between label l_k and some label in S_i

The dissimilarity between two labels is approximated by the distance measure between the corresponding categories in the class taxonomy. As seen in the formula above, the dissimilarity of one set of labels with respect to the other is taken as the average of the minimum distance between each label in the first set and some label in the second set:

$$dist(l_i,S_j) = \min_{l_k \in S_j} dist(l_i,l_k) \quad (108)$$

The semantic distance between two categories in the taxonomy approximates the amount of information in the categories' definitions by their depth in the taxonomy. The formula we used is:

$$(109)\quad dist(l_i,l_j) = \min_k \frac{bdist(a_k,l_i) + bdist(a_k,l_j)}{bdist(top,a_k) + bdist(a_k,l_i) + bdist(a_k,l_j)}$$

Where: l_i, l_j = the two labels
a_k = most specific common ancestors of the two labels
top = the top label in the taxonomy
bdist(a,l) = branch distance between ancestor *a* and label *l*

The branch distance between an ancestor category and a subconcept category is computed as the maximum distance between them over all possible branches that connect them.

7.2. Evaluation results

In the evaluation we used a subset of 700 explanations randomly chosen from a corpus of 2700 explanations collected during the same pilot study mentioned in Chapter 1. The category taxonomy consists of 198 categories.

Two human raters part of our research group labeled the set of 700 examples, assigning one or more labels to each explanation. Each rater went through the data twice, independent of one another, in an attempt to achieve maximum accuracy. For each of the three agreement measures, we computed the κs between two human raters, as well as the average of the κs between the system and each of the two human raters.

		κ	Actual Agreement	Chance Agreement	σ_κ
Set Equality					
	Human-Human	0.84	0.84	0.034	0.014
	System-Human	0.65	0.66	0.025	0.018
Overlap					
	Human-Human	0.87	0.88	0.040	0.012
	System-Human	0.73	0.74	0.033	0.016
Weighted Overlap					
	Human-Human	0.92	0.94	0.30	0.009
	System-Human	0.81	0.87	0.30	0.012

Table 7-1. Average pair-wise inter-rater agreement between human raters and average pair-wise agreement between the system and each human rater.

The results are shown in Table 7-1. Compared to our previous evaluation (Aleven & al, 2002), the results for the weighted overlap method show a decrease of the human-human agreement by about .02, and an increase in the average system-human agreement by about .03. So the gap between humans and the system decreased from .16 to .11.

The lower human-human agreement could mean that the classification task was harder this time around. That could happen either because the corpus was more difficult or because of the increased number of classes the raters had to choose from (about 200, versus about 150 in the last evaluation). The higher system-human agreement probably reflects the improved performance of the system.

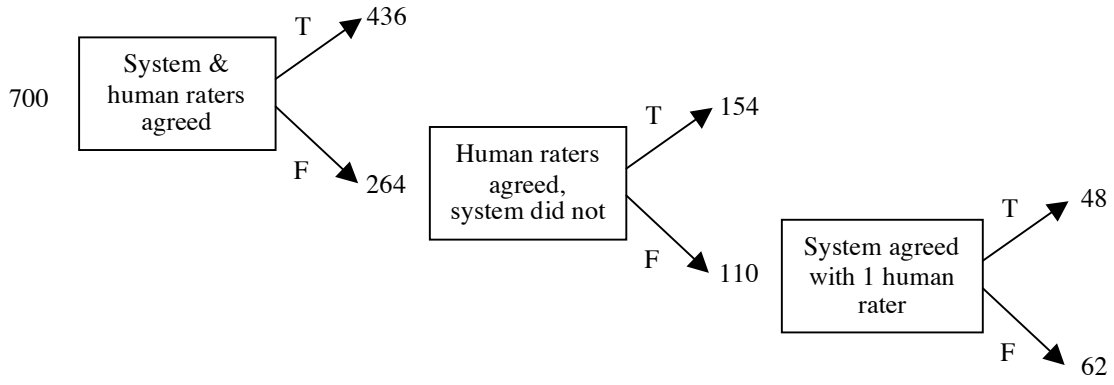

Figure 7-1. Agreement distribution among raters

A closer look at how agreement was distributed among the raters and the system reveals the picture in Figure 7-1. There are 436 cases (about 62%) where all raters agreed, 202 cases (154+48, about 29%) where two of the raters agreed and the third one did not, and 62 other cases (about 9%) where none of the raters agreed.

We performed a case-by-case examination of the 154 cases where the human raters agreed and the system did not, trying to reveal the cause for the system failure to generate the same set of labels as the humans. The examination showed there is no single big problem with the system, but rather a large collection of small problems, each of them affecting a couple of cases. We grouped these problems into several larger categories, listed below with examples:

- Robustness in the face of ungrammatical input:

(110) a) The measure of an angle formed by two adjacent angles is equal to the sum measures of the of the adjacent angles.

b) 2 angles that are next to each other have are the same.

- Syntactic or semantic coverage of new ways of expression:

(111) a) The angles sum up to because they are on a line and adjacent.

b) When a ray is drawn off part of a line, the angles on one side add up to 180.

c) Angle blu and angle ule are complementary angles so when added together their sum should equal 90 degrees.

- Problems with implicit references or ellipses:

(112) a) Subtract the measure of the vertical angle by the measure of the interior angle formed.

b) Three adjacent angles make up the larger angle so you must add them to find the measure.

- Bugs in various system components.

The study confirmed two things. First, that the system does an adequate job on capturing the meaning of students' explanations in an application that requires high accuracy. And second, that there are still an important number of improvements that can be made to the

system to increase its performance. One promising solution involves connecting semantic fragments of a partial parse structure through paths found in a limited directed search through the semantic space of the concepts involved.

An example of a real case dialog taken from our high school study where the system shows good performance was presented earlier in Table 1-1 in Chapter 1.

The dialog in Table 7-2 below, taken from the same study, shows an example where the NLU component of the tutor was not very successful in understanding the students explanations, as can be seen by comparing the system's classifications to the human raters' classifications. Despite this problem, the tutor may still be somewhat effective, since in this example the student is still able to get to the right explanation about Vertical Angles. However, the NLU system has again difficulty recognizing it as a correct explanation, because of problems with verb phrase coordination.

Part.	**Explanation/Feedback**	**System's Class**	**Humans' Class**
Student	angles across from one another are equal	(OPPOSITE–ANGLES REASON)	(CONGRUENT–OPPOSITE–ANGLES)
Tutor	And can you state a general rule about the kind of opposite angles that we have here? The tutor understood only the part of your explanation shown in red. You might change or delete the part SHOWN IN BLACK CAPITALS.		
Student	angles across from one another are equal	(OPPOSITE–ANGLES REASON)	(CONGRUENT–OPPOSITE–ANGLES)
Tutor	And can you state a general rule about the kind of opposite angles that we have here? The tutor understood only the part of your explanation shown in red. You might change or delete the part SHOWN IN BLACK CAPITALS.		
Student	angles across from one each other are equal	(REASON OPPOSITE–ANGLES)	(CONGRUENT–OPPOSITE–ANGLES)
Tutor	And can you state a general rule about the kind of opposite angles that we have here? The tutor understood only the part of your explanation shown in red. You might change or delete the part SHOWN IN BLACK CAPITALS.		
Student	angles in a rectangle across from one another are equal	(OPPOSITE–ANGLES RIGHT–ANGLES–OF–	(CONGRUENT–OPPOSITE–ANGLES

		RECTANGLE REASON)	ANGLES-OF-RECTANGLE)
Tutor	Yes, you are on the right track. One thing, though: this step does not involve a right angle. So it is better to delete or change this part.		
Student	these angles are	(ANGLES)	(ANGLES)
Tutor	It looks like you have the right idea. Can you state this geometry rule more completely?		
Student	these angles are vertical angles	(ANGLES VERTICAL-ANGLES)	(VERTICAL-ANGLES)
Tutor	Yes, and what about these types of angles?		
Student	these angles are vertical angles and are equal	(ANGLES VERTICAL-ANGLES REASON)	(CONGRUENT-VERTICAL-ANGLES)
Tutor	You are focusing on the right geometry rule. Can you state this rule? The tutor understood only the part of your explanation shown in red. You might change or delete the part SHOWN IN BLACK CAPITALS.		
Student	these angles are vertical angles and are equal	(ANGLES VERTICAL-ANGLES REASON)	(CONGRUENT-VERTICAL-ANGLES)
Tutor	You are focusing on the right geometry rule. Can you state this rule? The tutor understood only the part of your explanation shown in red. You might change or delete the part SHOWN IN BLACK CAPITALS.		

Table 7-2. Example of unsuccessful student-tutor dialog from the pilot study

7.3. Robustness issues

As discussed in the previous section, one of the problems revealed by the examination of failure cases is that the NLU system seems to have difficulties with ungrammatical input sentences. In order to better understand the system's behavior when dealing with various kinds of ill-formed input, we performed a more detailed analysis on the entire 700 sentences corpus.

In this analysis we examined each sentence in the corpus and labeled it according to whether it is a well-formed sentence or whether it exhibits some problems. We classified

the problems into 7 main categories, which can be either of a grammatical nature (5 categories), or semantic nature (2 categories). Some sentences were multiple labeled, when they presented several different problems. Overall, in the 700 sentence corpus there were 553 sentences that were judged well-formed and 147 sentences that were judged to show various problems.

As a first step in this analysis we computed the agreement κ between human raters, as well as between the system and the human raters, for the broad categories mentioned above: well-formed vs. ill-formed, as well as splitting the latter into grammatical and semantic problems. The results are seen in Table 7-3 below.

		κ all	κ well-formed	κ ill-formed	κ grammar	κ semantics
Number of Cases		700 (100%)	553 (79%)	147 (21%)	97 (14%)	57 (8%)
Set Equality						
	Human-Human	0.84	0.88	0.69	0.69	0.69
	System-Human	0.65	0.73	0.35	0.33	0.38
Overlap						
	Human-Human	0.87	0.90	0.77	0.77	0.74
	System-Human	0.73	0.80	0.45	0.42	0.49
Weighted Overlap						
	Human-Human	0.92	0.94	0.86	0.86	0.86
	System-Human	0.81	0.86	0.61	0.59	0.64

Table 7-3. Average inter-rater agreement between human raters and the system on problematic input

The results show that both human raters and the system perform better on well-formed sentences than on ill-formed sentences. On unproblematic input, the agreement between the system and the human raters improved on all three measures more than the agreement between human raters, reducing the distance to the latter by about .03-.04. As a result, it came within 0.1 range on the overlap measure, and within 0.8 range on the weighted overlap measure.

As the same time the drop in agreement scores on problematic input is much higher between the system and human raters than between human raters. The distance between the two increased by .14-.18 on the various measures. This result shows that human raters are significantly better than the system in dealing with problems in the input sentence, thus revealing an important area where the system could be improved.

Looking at the split between grammatical and semantic problems, we can observe that the system is more sensitive to grammatical problems, although not by much, the difference laying within .5-.7 on the 3 different measures. This difference shows that the weakest component when dealing with problems in the input stream is the parser-grammar tandem. This weakness is made more important by the fact that the number of grammatical problems was almost twice the number of semantic problems in our corpus.

This weakness is to be expected, since the parser-grammar formalism lacks a good mechanism to enforce constraints when needed and relax them when appropriate. The parser does provide a few such mechanisms, like marking specific features as “soft” or

skipping input words, but a try to apply them led to a serious increase in non-determinism and unacceptably long processing times, or even non-termination on some sentences. More focused mechanisms are needed, that relax constraints only in places where they are needed.

7.3.1. Grammatical problems

As mentioned before, the two broad categories of problems were split into several more refined categories, which might require different solutions. The 5 grammatical categories we found are presented in the following sections. As the previous table shows, we had a total of 97 sentences with grammatical problems.

7.3.1.1. Agreement (23 cases)

These are cases where there is a number disagreement between two sentence elements. Most often, that involves subject-verb agreement, but there are also some cases of determiner-noun agreement. A few examples are:

(113) a) two angles that make up a linear pair is equal to 180 degrees

b) adjacent angles can add up to one adjacent angles

c) sum of all angle add up to 180

7.3.1.2. Extra words (10 cases)

These are cases when a word is present in the input sentence that should be removed to make the sentence grammatical. Some times this is just some punctuation mark, like in example (114) a) below, other times it is an actual word:

(114) a) the shape is an isosceles triangle so the two bottom angles,, are equal

b) 2 angles that are next to each other have are the same

c) if 2 parallel lines are intersected by a transversal, interior angles are the supplementary

7.3.1.3. Missing words (27 cases)

A larger category is represented by sentences where words are missing from the input stream. There is a large variation in this category, from cases where only some particle is missing, like a possessive marker (see example (115) a) below) or a preposition (example (115) b)), going through cases where a content word is missing, like a noun (examples (115) c) and d)), to cases where an entire predicate argument, like a predicate or an object (examples (115) e) and f)), is missing:

(115) a) sum of all triangle 3 interior angles add up to 180

b) the two adjacent angles add 180

c) the measure of an adjacent is equal to the sum of those angles

d) the angles sum up to because they are on a line and adjacent

e) vertical angles are

f) angle ste added to angle etp make

7.3.1.4. Wrong words (38 cases)

An even larger category of problems in our test data is given by sentences where one word should be replaced by another one or two in order to have a grammatical sentence. So such cases can be considered as a combination of the previous two cases, where a word is extra and a few are missing. Some times the problem is just an extra space, like in example (116) a), other times it is just a functional word, like in cases (116) b) and c), but other cases, like (116) d) and e) are more complex. Some of these are cases when the spelling checking process did not choose the best solution.

(116) a) two angles with in an angle added together equal that angle

b) in a triangle their is 180 degrees

c) the angles are supplementary and they form a line so they add up two 180

d) when 2 numbers are complementary, they add told

e) these angles are complement two angles which the sum is equal to 90 degrees

7.3.1.5. Wrong references (5 cases)

A small number of cases was that of sentences where a referent does not seem to have a good antecedent. This can happen either because of a number mismatch, as in examples (117) a) and b) below, or because the antecedent just seems to be missing, like in the next example.

(117) a) the sum of the two adjacent angles must total 180 because it is a supplementary angle

b) the linear angles must total 180 then added together because it formed a supplementary angle

c) the measure of an adjacent is equal to the sum of those angles

7.3.1.6. Robustness results on grammatical problems

The κ agreement scores for the various subcategories of grammatical problems discussed above are shown in Table 7-4 below.

		κ grammar	κ agreement	κ extra words	κ missing words	κ wrong words	κ wrong references
Number of Cases		97 (100%)	23 (24%)	10 (10%)	27 (28%)	38 (39%)	5 (5%)
Set Equality							
	Human-Human	0.69	0.81	0.67	0.61	0.62	0.52
	System-Human	0.33	0.46	0.05	0.46	0.23	0.00
Overlap							
	Human-Human	0.77	0.84	0.72	0.71	0.73	0.67
	System-Human	0.42	0.52	0.10	0.50	0.41	0.10
Weighted Overlap							
	Human-Human	0.86	0.90	0.86	0.79	0.85	0.77
	System-Human	0.59	0.66	0.36	0.69	0.52	0.25

Table 7-4. Average inter-rater agreement between human raters and the system on ungrammatical input

The table shows a significant difference between the system and human raters in how they are able to cope with various kinds of problems. The human raters did not show too much variation in their ability to deal with these different categories, in the weighted overlap measure for instance the κ varying between 0.90 and 0.77. The system displayed much more unequal ability to deal with these problems, the κ in the similar case varying from 0.69 to 0.25. In order to better compare the system to human raters over the various categories we defined a system-human performance ratio as:

$$r_p = \frac{\kappa_{SH}}{\kappa_{HH}} \quad (118)$$

Where: κ_{SH} = the average system-human agreement
κ_{HH} = the average human-human agreement

	Number of Cases	Avg. Human-Human κ	Avg. System-Human κ	System-Human Performance Ratio	Failure Frequency
Grammar	97	0.86	0.59	0.69	30
Agreement	23	0.90	0.66	0.73	6
Extra words	10	0.86	0.36	0.41	6
Missing words	27	0.79	0.69	0.87	4
Wrong words	38	0.85	0.52	0.62	15
Wrong references	5	0.77	0.25	0.32	3

Table 7-5. Performance ratio and failure frequency on ungrammatical input

The results for this ratio on the weighted overlap measure can be found in the fifth column of Table 7-5. They show that the system is able to deal much better with two of the categories, those of missing words and agreement, while at the same time having

more serious problems with two other categories, those of extra words and wrong references.

The two categories where the system performed worst were however also the ones that were the least frequent in our corpus. In order to measure the impact of these difference categories of problems on the system performance our corpus we computed a failure frequency measure, defined as:

(119) $f_f = n \cdot (1 - r_p)$

Where: n = number of cases
r_p = system-human performance ratio

The corresponding values for this measure, found in column 6 of Table 7-5 show that indeed a different category, that of wrong words, has the highest impact on system performance, contributing to about half of all grammatical failures.

7.3.2. Semantic problems

The second broad category of problems in input sentences in our corpus is that of semantic problems. They are sentences that are grammatical in structure, but there is a mismatch in the meaning of various parts that make the whole meaning violate some semantic constraints. There are a total of 57 such cases in our corpus. They were only split in two subclasses, discussed below.

7.3.2.1. Number problems (17 cases)

These are cases when the number agreement between various parts of the sentence is correct. However the number used for some components is in contradiction with the properties asserted on those components. A few examples are given below.

(120) a) the sum of a supplementary angle is 180

b) in an isosceles triangle, angles opposite congruent side are congruent

c) the measure of the angle formed by an adjacent angle is equal to the total

7.3.2.2. Proper semantic problems (40 cases)

The last category is a catch-all for all other semantic problems that did not involve number problems. Because of the highly heterogeneous nature of these cases it was difficult to further group them into classes. Sometimes the problem was caused by a small functional word, like in cases (121) a) and b) below. Most cases involved more complex mismatches.

(121) a) the sum of the angles in a line make 180 degrees
b) base angles are congruent on an isosceles triangle
c) angles are the same when intersected by two lines
d) isosceles triangles have isosceles angles
e) the angle is a isosceles triangle
f) the sum of isosceles angles is 180

7.3.2.3. Robustness results on semantic problems

		κ semantic	κ number problems	κ proper semantic problems
Number of Cases		57 (100%)	17 (30%)	40 (70%)
Set Equality				
	Human-Human	0.69	0.68	0.69
	System-Human	0.38	0.62	0.28
Overlap				
	Human-Human	0.74	0.71	0.74
	System-Human	0.49	0.74	0.37
Weighted Overlap				
	Human-Human	0.86	0.85	0.86
	System-Human	0.64	0.84	0.56

Table 7-6. Average inter-rater agreement between human raters and the system on semantically ill-formed input

Table 7-6 above gives the agreement scores for the three measures on sentences with semantic problems. Again the table shows a serious difference on the two subcategories between human raters and the system. Humans are affected about the same in their ability to judge sentences by the two kinds of problems. On the other hand the system has no issue with semantic number problems, but its performance is seriously affected by the rest of the semantic problems.

	Number of Cases	Avg. Human-Human κ	Avg. System-Human κ	System-Human Performance Ratio	Failure Frequency
Semantic	57	0.86	0.64	0.75	14
Number problems	17	0.85	0.84	1.00	0
Proper semantic problems	40	0.86	0.56	0.65	14

Table 7-7. Performance ratio and failure frequency on semantically ill-formed input

Again we computed the performance ratio and the failure frequency in these cases for the weighted overlap measure. The values in Table 7-7 above show the same conclusion. At the same time, comparing these values with the similar values on grammatical problem sentences we can notice while the system-human performance ratio is much higher, the

failure frequency is still about half that resulted from grammatical problems. This is due to the presence of an important number of such cases in out corpus.

7.4. Comparison of performance with a Naïve Bayes approach

Although the performance level reflected by the classification results in section Chapter 7 looks pretty good, there is a possibility that the classification problem is not very difficult, so it could be performed to the same level of performance with simpler, already existing methods. In order to see whether this is the case, we conducted a study with the Naïve Bayes approach.

There are two characteristics of the data we had available that makes the implementation of the Naïve Bayes approach more problematic. First, we had two sets of labels for each sentence, coming from two different human raters, with no agreement between the two sets on a correct labeling. And second, each sentence was labeled with a set of labels that cover the whole meaning, while Naïve Bayes normally works with single labels.

The first problem was dealt with by running the test twice, once for each set of labels. In the two runs the system was first trained and then tested using the same set of labels, coming from one of the two raters.

The second problem was solved by treating sets of labels associated with an explanation as a single label. The sets were thus used both to train the classifier and to test it. This approach has the negative effect to make the sparseness of the data problem more acute. Using this technique, the number of distinct labels has increased from about 100 to about 150, which was deemed to remain within reasonable limits.

An alternative was considered, which is to use each of the labels in a set for training separately. However in this case the result from the classifier would also consist of one or more single labels for each sentence, each with a different score. It is not clear how to compare these results with the original set of labels. If only the label with the best score is considered in the test, the results would probably be poor, since most sets have at least two labels. If we take several labels for comparison, it was not clear how to decide how many of them to take into account. Besides, there is a good chance that only one of the labels with the higher scores is correct, as they are generated as alternative single labels for the sentence, not complementary partial ones.

The test was conducted on the same set of 700 explanations used in the evaluation of the system's NLU performance. To avoid the over-fitness problem, a 10-fold cross validation was employed. In each run the classifier was trained on 9/10 of the explanations chosen randomly from the whole set, and tested on the remaining 1/10. The same three κ statistic measures were used as in the NLU evaluation. The results are presented in Table 7-8 (the Average NB-Human lines) together with our NLU system results (the Average NLU-Human lines), for easy comparison.

		κ	Actual Agreement	Chance Agreement	σ_κ
Set Equality					
	Human-Human	0.84	0.84	0.034	0.014
	Average NLU-Human	0.65	0.66	0.025	0.018
	Average NB-Human	0.42	0.45	0.062	0.020
Overlap					
	Human-Human	0.87	0.88	0.040	0.012
	Average NLU-Human	0.73	0.74	0.033	0.016
	Average NB-Human	0.44	0.47	0.065	0.020
Weighted Overlap					
	Human-Human	0.92	0.94	0.30	0.009
	Average NLU-Human	0.81	0.87	0.30	0.012
	Average NB-Human	0.63	0.75	0.33	0.016

Table 7-8. Average system-human agreement between Naive Bayes and two human raters

The table shows a gap varying from about 0.23 for the set equality measure, to about 0.29 for the overlap measure, and to about 0.18 for the weighted overlap measure. The gap is large enough to demonstrate the NLU system performs considerably better than a simple statistical classification method like Naïve Bayes.

One problem the above result does not address is whether we have enough data for the Naïve Bayes method, or whether giving it more data would significantly improve the results. In order to find that, we performed the same test as above 9 times, each time giving the method only a fraction of the data for training (going thus from 1/10 to 9/10), and each time testing it on the 1/10 of data set aside.

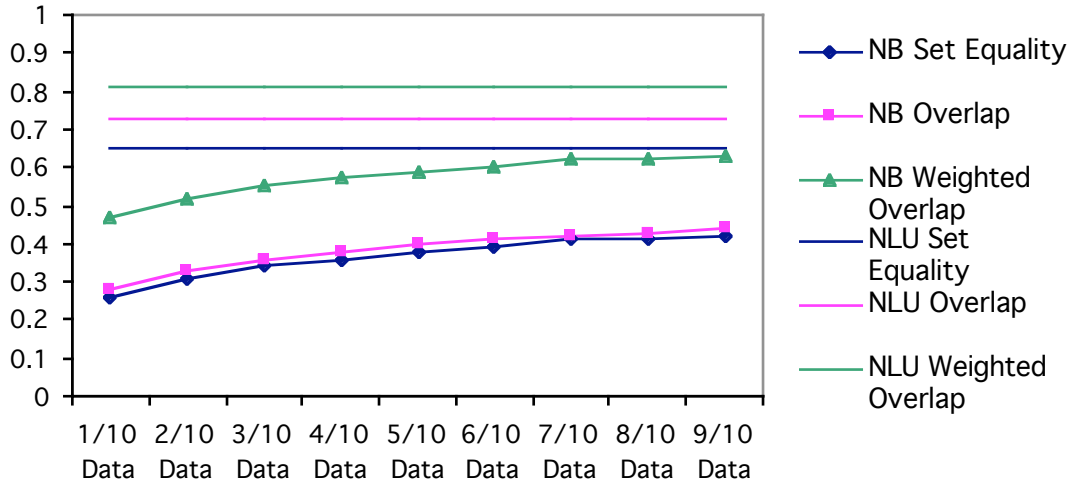

Figure 7-2. Naïve Bayes agreement when varying the amount of training data

The results are shown in Table 7-9 below and are graphed in Figure 6-2 above. The chart shows that for the higher amounts of training data the improvement in classification agreement is minimal from one step to the next. Thus, it seems reasonable to conclude that the Naïve Bayes method is in the saturation area, and more training data would not help it much more.

		κ	Actual Agreement	Chance Agreement	σ_κ
Set Equality					
	1/10 Data	0.26	0.31	0.064	0.019
	2/10 Data	0.31	0.36	0.067	0.019
	3/10 Data	0.34	0.39	0.067	0.020
	4/10 Data	0.36	0.40	0.066	0.020
	5/10 Data	0.38	0.42	0.065	0.020
	6/10 Data	0.39	0.43	0.064	0.020
	7/10 Data	0.41	0.45	0.063	0.020
	8/10 Data	0.41	0.45	0.062	0.020
	9/10 Data	0.42	0.45	0.062	0.020
Overlap					
	1/10 Data	0.28	0.33	0.068	0.019
	2/10 Data	0.33	0.38	0.070	0.019
	3/10 Data	0.36	0.40	0.070	0.020
	4/10 Data	0.38	0.42	0.069	0.020
	5/10 Data	0.40	0.44	0.068	0.020
	6/10 Data	0.41	0.45	0.067	0.020
	7/10 Data	0.42	0.46	0.066	0.020
	8/10 Data	0.43	0.47	0.065	0.020
	9/10 Data	0.44	0.47	0.065	0.020
Weighted Overlap					
	1/10 Data	0.47	0.65	0.34	0.017
	2/10 Data	0.52	0.68	0.34	0.017
	3/10 Data	0.55	0.70	0.33	0.017
	4/10 Data	0.57	0.71	0.33	0.016
	5/10 Data	0.59	0.73	0.33	0.016
	6/10 Data	0.60	0.73	0.33	0.016
	7/10 Data	0.62	0.74	0.33	0.016
	8/10 Data	0.62	0.75	0.33	0.016
	9/10 Data	0.63	0.75	0.33	0.016

Table 7-9. Average agreement between Naïve Bayes and human raters when varying the amount of training data

7.5. Comparison of performance with K-Nearest Neighbor approach

Even if the Naïve Bayes method seems to have reached its full potential, it is not considered one of the most successful classification methods to date, although it seems to

be one of the most robust methods. In order to have a better understanding of how our results compare with statistical classifiers, we compared it against a second approach, which is reported to give consistently better results, the k-nearest neighbor (Mitchell, 1997).

For k-nearest neighbor implementation we used the Weka package (Witten & Frank, 2000) that is freely available online. We converted the data into the required input format, where each sentence has as symbolic attributes the set of classes from the human rating and a yes/no value for each word in the lexicon signaling whether it was present in the sentence or not.

Similarly to the Naïve Bayes case, for training purposes we treated each set of labels as being a single label, and we dealt with the two different sets of human labels by running the training and the test twice, once on each set, and taking an average.

The test was run on the same set of 700 sentences used to test the performance of our system, and again we used 10-fold cross-validation to avoid over-fitting. We also run a preliminary test for different values of K, to see which one works better. The best results were obtained for K=1.

		κ	**Actual Agreement**	**Chance Agreement**	σ_κ
Set Equality					
	Human-Human	0.84	0.84	0.034	0.014
	Average NLU-Human	0.65	0.66	0.025	0.018
	Average KNN-Human	0.55	0.57	0.038	0.020
Overlap					
	Human-Human	0.87	0.88	0.040	0.012
	Average NLU-Human	0.73	0.74	0.033	0.016
	Average KNN-Human	0.60	0.62	0.043	0.019
Weighted Overlap					
	Human-Human	0.92	0.94	0.30	0.009
	Average NLU-Human	0.81	0.87	0.30	0.012
	Average KNN-Human	0.76	0.83	0.31	0.013

Table 7-10. Average system-human agreement between K-Nearest Neighbor and two human raters

The results are shown in Table 7-10 above, in comparison with our NLU system. As it can be seen, the k-nearest neighbor approach gives results that are closer to our system, but our NLU system still maintains a statistically significant advantage.

Chapter 8 Applying Logic-Based NLU to the Algebra Domain

One of the claimed advantages of logic-based NLU versus a semantic grammar approach is that because of the separation of syntax and semantics, our approach would be easier to extend to new domains. To verify this claim, we implemented the same approach in a new semantic domain.

8.1. Explanation in equation solving

The domain chosen for the new application is in the context of an Algebra tutor that teaches students how to solve linear equations. A tutoring strategy that has potential to improve students' skills at solving those equations is to ask them to explain the steps they take when solving the equations, as was done in Geometry (Aleven & al, 2001). Some possible examples of such explanations are:

(122) a) I added 10 to both sides of the equation to get rid of the –10.

b) When I add the same thing to both sides, the equation is still true.

In this context, similar to the geometry tutor, the tutoring system would classify student's explanations and give them feedback on the correctness and completeness of their explanations.

8.1.1. Evaluation of development effort

The main goal of the new implementation was to evaluate the amount of work needed to bring the Algebra NLU application to a level of performance similar to that of the Geometry application. This evaluation can be done in two ways: First, we counted the number of new elements that need to be added to the system to implement the new application. And second, we measured the total time spent on developing the new application, as well as the relative proportion of time spent on various components.

The expectation before the process of implementing the new domain application started was that most of the system's components can be reused with no or minimal changes. Thus we expected to reuse:

- All basic mechanisms like LCFlex and Loom
- Most of the Linguistic Inference Module

- The entire grammar, with minor adjustments
- The Upper Model of the ontology.

On the other hand, we expected to have to reimplement components that model the knowledge in the new domain:

- Lexicon
- Domain-dependent ontology
- Classification hierarchy.

The expectation in terms of time was that the new application would take a couple of months of effort, with most of the time spent on modeling the new domain knowledge.

8.1.2. Evaluation of data needs

A second goal of the new implementation was to test another potential benefit of our approach versus statistical approaches: developing new applications without the need of large corpora. In order to verify this issue, the development work was done in two stages. First a small corpus of 57 sentences was collected from two high school textbooks (24) and from two of the system developers (33). This corpus was used to implement a preliminary version of the Algebra application.

In a second stage, a larger corpus of 338 explanations was collected from colleagues in the PACT group (which includes research scientists, research programmers, and graduate students). This corpus was used first for a preliminary test, and second for further development to the system. To verify that the Algebra application is at about the same level of performance as the Geometry application, a final test was performed at the end of the development process using a third corpus of 517 explanations collected online.

Unlike the development corpus, the test corpus was collected from CMU undergraduate students. This difference could potentially make the test more difficult, since the testing data could show significant linguistic differences versus the training data. The results in section 8.3 show that the developed system is not affected negatively by such differences.

8.2. Modifications required by the new application

The data collected both in the first and in the second development stage shows that the language used in the Algebra domain is indeed sufficiently different form the language used in the Geometry domain to warrant calling the new implementation a new semantic domain. All modules that model the new domain needed serious reconstruction.

8.2.1. Lexicon additions

As expected, the Algebra vocabulary and concepts were mostly new. Some typical sentences:

(123) a) `Break down fractions on the lhs and rhs so that you can combine like terms later.`

b) `Multiply both sides by 2 to get rid off the fractions on the lhs.`

c) `To get rid of the fraction on the rhs, divide numerator and denominator by 3 then divide the constant 10 by 2.`

In the examples above, most content words are new. Even when words are common with the Geometry application, their meaning is usually different. For example, '`side`' is used in Algebra to denote the 2 sides of an equation. In Geometry, '`side`' was used with two different meanings: '`side of a triangle`', and '`side of a transversal`'. The second meaning comes closer to the meaning in Algebra, but is still not the same, since the sides of an equation are algebraic formulas, while the sides of a transversal are plane areas. All new word meanings needed new lexicon entries to reflect the new meanings.

The system lexicon is split into two components: a domain independent lexicon common to all applications and a domain specific lexicon. The domain independent lexicon came with 198 entries from the Geometry application. The Algebra application required an addition of 28 new entries to the domain independent lexicon, mostly prepositions, adjectives, and nouns. At the same time, 9 entries (mostly nouns and verbs) were moved from the domain independent lexicon to the Geometry lexicon, because the corresponding concept definitions were specific to the Geometry context. Thus, the domain independent lexicon reached 217 entries.

The Algebra specific lexicon was developed from scratch and over the two development phases it reached a total of 88 entries of three different parts of speech: adjectives, nouns, and verbs. New words with other parts of speech were made part of the domain independent lexicon because they were judged not specific to the Algebra semantic domain. The size is comparable with the Geometry lexicon, which contains 105 entries (after the move).

Overall, 189 entries out of a total of 294 in the Geometry application have been reused, giving a reuse percentage of 64%. At the same time 116 entries, or a percentage of 38% out of a total of 305 in the Algebra application are new. The results are summarized in Table 8-1 in section 8.2.6 below.

8.2.2. Knowledge base additions

The new vocabulary needed also new concepts. Following the lexicon, the system's knowledge base is also divided into two sections: a domain independent section and a domain specific section. The domain independent section, called the Upper Model, could be used mostly as it resulted from the development process in the Geometry application. Two concepts out of the total of 205 were moved to the Geometry specific application, following the move of the corresponding lexicon entries. A number of 15 concepts that deal with the new words in the domain independent lexicon were added to the Upper Model. Two new production rules were needed to deal with inferences required by the new domain. Thus, the Upper Model reached a total of 220 elements.

The Algebra domain needed the creation of 73 new elements: 61 concepts, 1 relation, and 11 production rules. This number is significantly lower than that for Geometry, which totals 179 elements after the move. The difference seems to reflect a somewhat lower conceptual complexity of the Algebra domain.

The numbers above lead to a reusability percentage for the knowledge base of 53% (203 out of 381), and a percentage of new elements of 30% (88 out of 293). The resulting knowledge base is structured in three parts, as Figure 8-1 below illustrates: a common Upper Model knowledge base, and two domain-specific knowledge bases.

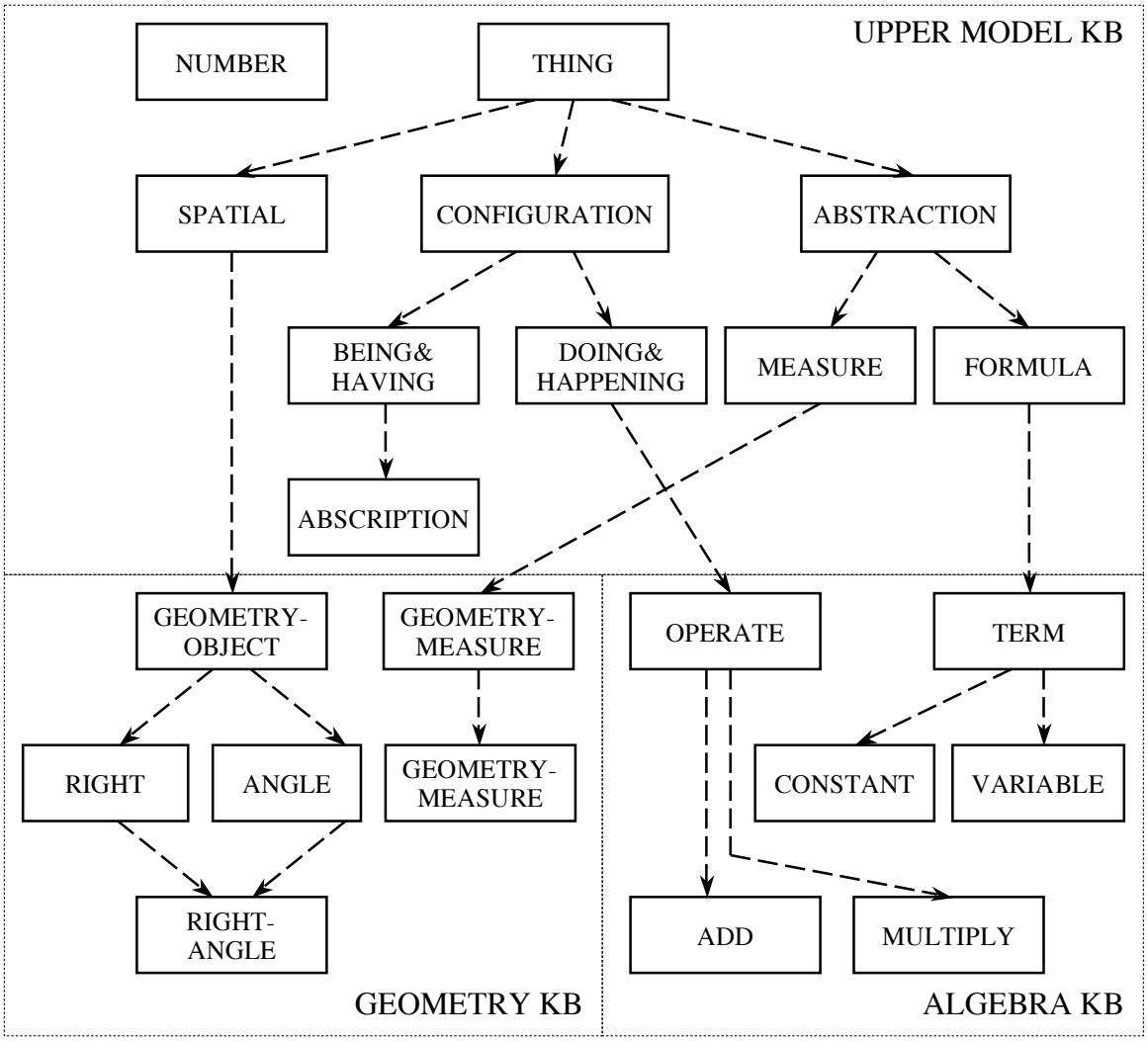

Figure 8-1. Example of partial Upper Model with Geometry and Algebra concept hierarchy

8.2.3. Metonymy resolution

The Linguistic Inference module showed the same pattern as the rest of the system in terms of changes needed for the Algebra application. All general combination rules

remained unchanged. Similarly, the reference resolution mechanism could be reused without change. The metonymy resolution rules however needed to be updated for the new application. This is to be expected, since metonymy is more semantic and domain dependent than the other mechanisms in the Linguistic Inference module.

One of the two metonymy rules used in the Geometry application was met again in Algebra, although indirectly. It involves cases where one of the elements that need to be combined is a set. An example from the Geometry application can be seen below:

(124) Sum of a linear pair must equal 180 degrees.

Here the problem consists of the fact that the 'sum' is not the sum of the 'pair', as the text explicitly asserts, but rather the sum of the unspecified elements of the pair, which are implied to be linear angles.

Unlike in Geometry, the set in Algebra is usually not expressed directly in the language, but rather results from dealing with coordination. Thus, it cannot be called metonymy, although the same inference rule applies:

(125) Execute subtraction of both the variable and constant terms.

Here 'variable and constant' is an adjective coordination, which is modeled in the logic language by a set. The metonymy rule for sets is applied when this set is combined with 'terms'. It performs the combination by distributing the meaning of 'terms' over the two generic elements of the set, thus resulting in a semantic structure equivalent to the expression 'variable terms and constant terms'.

The metonymy case that was met quite often was specific to the Algebra domain. It involves using the term 'side' with the meaning of 'side of an equation' to stand for the algebraic expression that is located on the respective side.

(126) a) Divide both sides by the multiplicative constant 3 to isolate x.

b) Add 3x to both sides to isolate x terms on the left.

c) Subtract 17 from both sides to zero out constant on the left and isolate constant terms on the right.

d) You subtract one of the x terms from both sides so you have only one x term on one side of the equation.

'Side' cannot be modeled as an element that is part of an algebraic operation because there are cases, like in the phrase 'one x term on one side' in example (126) d) above, when it is used to explicitly specify the location of another term. Modeling 'side' as both an expression and a location of an expression would make impossible to set semantic constraints in cases when only one of them is adequate. Thus, it has to be modeled as only a location specification. Then, in order to get the correct semantic representation for cases like (126) a) metonymy resolution rules are needed in order to assert the existence of an actual expression to be involved in the operation, related through a 'location' relation to the 'side' in question.

8.2.4. Classification taxonomy

As expected, the new Algebra application needed a completely new classification taxonomy. A detailed examination of the approximately 400 sentences in our development corpora led to the creation of a number of 208 category definitions, close to the 220 categories used in the Geometry application. All the categories are new, which was to be expected given their highly application-dependent nature.

Since a full tutor was not developed in conjunction with the Algebra application, there is some uncertainty about how many of these classes would be useful in an actual tutorial environment, or if more classes would need to be added to model our corpora. The decision depends on what aspects of the language semantics are considered to be relevant for the tutorial process.

A number of these classes were developed by symmetry, even if they did not model any of the sentences in our corpora directly. For instance, any time an action was found to be asserted on one of the two sides of the equation, a similar class was created for the other side.

(127) `Combine variable terms on the left side.`

For example, the explanation in (127) needed a class called `COMBINE-VARIABLES-LEFT-SIDE`. By symmetry, another class called `COMBINE-VARIABLES-RIGHT-SIDE` was also added to the hierarchy, even if no such example occurred in our development corpora.

Even if it is true that the semantic definitions of the two classes differ, it is not clear whether the difference is relevant for the tutorial process, or whether the two classes could rather be replaced with a more general class that does not make a distinction about what side the operation is performed on.

An interesting aspect that became apparent during the development of the classification hierarchy is that, similarly to the Geometry application, many of the classes are very close to each other in the semantic space. This characteristic makes the classification task more difficult for a statistical-based approach.

For example, there are two groups of classes, one expressing operations done among the elements of one side of the equation, the other expressing operations performed identically on both sides. The distinction between the two groups is important for an equation-solving tutor, as the two classes reflect distinct steps in the solving process. Examples of the two classes are `ADD-CONSTANTS-BOTH-SIDES` and `ADD-FORMULA-BOTH-SIDES`. The first class covers explanations like that in (128) a), while the second covers examples like (128) b).

(128) a) Add the constants on each side of the equation to simplify them.

b) Add 5 to both sides of the equation.

c) Add the same constant to both sides of the equation.

d) Add constants on both sides of the equation.

e) Add a constant to both sides of the equation.

f) Add the same constants on both sides of the equation.

However, constants are also a specific kind of formulas, so it cannot be used in the distinction, since an example like (128) c) should still classify as ADD-FORMULA-BOTH-SIDES. The expression 'each side' in (128) a) is not relevant for the distinction either, since example (128) d) still needs to classify under the first class. Neither is the use of preposition 'on' versus 'to', since example (128) f) needs to be classified under the first class, same as (128) c). What is relevant is a combination of factors that determine whether the elements added to the two sides are the same or not. This aspect can be expressed in various ways, either explicitly, using for instance 'same', like in (128) c) and f), or implicitly, through the use of a singular, like 'a constant' in (128) e). The use of singular is not relevant by itself, since number information could be overridden by the presence of an explicit element, like 'same constants' in example (128) f).

8.2.5. Grammar modifications

A less expected aspect of the new application was the fact that the Algebra language also showed new syntactic patterns. The root of the difference comes from the different nature of the configurations described in the new domain. In Geometry, as one can see in examples collected from our test corpus below, most of the sentences describe static configurations or relations present in the problem diagram.

(129) a) The measure of an angle formed by adjacent angles is equal to the sum of the adjacent angles.

b) An angle's measure is equal to the sum of the two adjacent angles that form it.

c) The sum of the measures of two adjacent angles is equal to the measure of the angle formed by the two angles.

d) When parallel lines are intersected by a transversal the same side angles formed are supplementary.

e) Two parallel lines intersected by a transversal have supplementary same side angles.

f) Adjacent angles that form a line equal 180 degrees.

g) All angles of a triangle add up to 180.

h) They are corresponding angles that are the same.

The verbs are most often 'be (equal)', 'equal', or 'have', which denote relations. Even when they are different, like 'add' or 'form', those are used in the intransitive form, to describe static configurations, not to denote actual actions performed by real

actors. One result of this characteristic is that very often parts of the configuration are expressed as a relative clause, as most sentences in the example above show.

On the contrary, most equation solving explanations describe actual actions taken during the process of solving an equation. As such, they present a variety of verbs denoting those actions, verbs that can take several different arguments: '`break down`', '`combine`', '`get rid`', '`multiply`', etc. As we can see in example (123) above, relative clauses are practically nonexistent, the information content being present mostly in the various verb arguments. Most verbs are in impersonal imperative form.

Coordination of verbs and/or arguments is also used more extensively in the Algebra application. The examples in (130) show various new forms of coordination met in the Algebra corpus. Example a) illustrates coordination of nouns that determine a base noun. Examples b) and c) show coordination of formulas sharing the same determiner. In example d) two coordinated verbs share arguments. Examples e), f), and g) show a trickier case where pairs of arguments are coordinated, while sharing the same verb.

(130) a) `Multiply both sides by -7 to prepare to isolate [[x and constant] terms] on opposite sides of the equation.`

b) `You can apply the distributive rule to [the [quantity x+5, x-1 and x+2]].`

c) `Distribute [the [2, 8, and 7]] across their products.`

d) `Get the variable by [[multiplying or dividing] the same constant] on both sides.`

e) `Prepare to isolate [x terms on one side of the equation] and [constant terms on the other side].`

f) `Distribute [1/3 on the left] and [-1/4 on the right].`

g) `Subtract [7x from 10x] and [7x from 7x].`

One other new aspect of the language in the Algebra application is the presence of formulas.

(131) a) `You can apply the distributive rule to the quantity x+5, x-1 and x+2.`

b) `Subtract 7x from 10x and 7x from 7x. This reduces the complexity of the equation and groups the x terms onto one side.`

c) `Get rid off -15x/6 on rhs by adding -15x/6 to both sides.`

In Algebra equation solving, knowing what formulas the operations are performed upon can be relevant to the correctness of the explanation. As such, rules for parsing formulas and integrating them with regular noun phrases are needed in the system grammar.

All these differences mean that different parts of the grammar are stressed in the new application, some of which were not fully developed before.

In order to implement these changes in syntax patterns, the grammar needed the addition of 24 rules. At the same time 4 rules were removed from the Geometry grammar because they were dealing with explanation labels and unit measure expressions, neither of which

are present in the Algebra application. Thus the grammar reached a reusability percentage of 97%, bringing the total from 135 used in the Geometry application to 155, and a renewal rate of 15%.

Another 41 rules needed adjustments or additions in feature unification equations to accommodate the new rules and adapt to other patterns.

In terms of number of feature structure equations, the grammar moved from 1791 to 2041. Out of the original 1791 equations, 66 have been modified, and 73 have been deleted, giving a reuse percentage of 96%. At the same time 322 equations, or 16%, have been added to the grammar.

8.2.6. Summary of modifications

A summary of the changes performed in the new application is given in Table 8-1 below. Overall, the structure of the changes is close to what was expected, with the domain-specific components needing most of the changes. The unexpected result that grammar rules and feature structures also needed some amount of development can be explained by the shift in syntactic patterns that also occurred in the Algebra application.

8.3. Evaluation of the classification accuracy of the new application

As discussed earlier, the development work was done in two stages. The first stage was based on a small corpus of 57 sentences collected from two high-school Algebra textbooks and two of the system developers. During this stage, which took about 60% of the total development time (or about 46 hours), the main vocabulary was added, as well as most of the concepts and of the classification categories.

	Elements in Geometry application	**Elements reused in Algebra application**	**Elements in Algebra application**	**Elements added in Algebra application**
Lexicon	294	189 (64%)	305	116 (38%)
Grammar rules	135	131 (97%)	155	24 (15%)
Feature structure equations	1791	1717 (96%)	2041	322 (16%)
KB elements	381	203 (53%)	293	88 (30%)
Classification categories	220	0 (0%)	208	208 (100%)

Table 8-1. Summary of changes in new application

At the end of the first development stage a larger corpus of 338 explanations was collected from the members of the PACT group. This corpus was labeled by hand using the existing categories, and was used for two purposes. First, a preliminary classification test was conducted with the NLU system on the labeled corpus. And second, the labeled explanations were used as further training data in the second development stage.

The classification agreement between the system and the human rater (the author) for the preliminary test are given in Table 8-2 below.

	κ	Actual Agreement	Chance Agreement	σ_κ
Set equality	0.43	0.44	0.019	0.027
Overlap	0.57	0.59	0.037	0.024
Weighted overlap	0.64	0.66	0.065	0.022

Table 8-2. System-human classification agreement at the end of the first development stage (using only 57 explanations) on the 338 explanation corpus

While the results are significantly higher than chance agreement, showing that real progress has been achieved, they are also significantly lower (about 0.2) than similar results in the Geometry application. Thus, a second development stage proved to be necessary.

It the second stage, the labeled test corpus was taken as the target for system classification. Modifications to the system mainly involved refining all system components with the explicit goal to maximize the classification accuracy. At the end of the stage the system was tested again on the development corpus, with the results seen in Table 8-3.

	κ	Actual Agreement	Chance Agreement	σ_κ
Set equality	0.80	0.81	0.015	0.022
Overlap	0.89	0.89	0.042	0.015
Weighted Overlap	0.91	0.91	0.080	0.013

Table 8-3. System-human classification agreement at the end of the second development stage (using 395 explanations) on the 338 explanation corpus

This time, the results are above the similar results in the Geometry application, suggesting that a satisfactory development level has been achieved. However the results are not very conclusive, since the same corpus has been used both for testing and for development. A test with a new corpus was needed to validate the above results.

In order to have a valid test, a new corpus of 517 sentences was collected by soliciting online participation of CMU students. This corpus was again labeled by hand, and then used for the final test of the system. The results can be seen in Table 8-4.

	κ	Actual Agreement	Chance Agreement	σ_κ
Set equality	0.77	0.78	0.036	0.019
Overlap	0.85	0.86	0.061	0.014
Weighted overlap	0.89	0.90	0.087	0.012

Table 8-4. System-human classification agreement at the end of the second development stage (using 395 explanations) on the 517 explanation corpus

The results are only slightly lower than those on the development corpus, thus proving conclusively that the Algebra application is developed to a level of performance comparable to the Geometry application.

8.4. Evaluation of time effort

The goals of measuring the development time the Algebra application were to get an accurate evaluation of the total time spent in development, as well as an evaluation of time percentage spent in developing the various parts of the system. At the same time, the process of measuring the time had to be done in such a way as to not interfere with the development process.

Previous attempts at measuring development time (Langley, 2003) did not seem adequate for the declared goals. Requiring the developers to simply write down the time spent on the system has problems. First, it relies on the developers remembering to write down the times accurately. And second, if the records have to be updated too often, like every couple of minutes, the process will interfere with the system development.

Two automatic time recording methods have been explored. First we tried to use software to record all activity on the computer screen, accompanied by audio commentary recorded by the developer. The audio/video recording needs to be examined and "coded" according to the desired goals in a second phase. This method proved to have serious problems. Although it did record all activity accurately, the audio recording proved to be more difficult, since it interfered with the development process. More importantly, in a test session, the "coding" phase proved to require an enormous amount of time, and thus it was not feasible.

A second method was then explored. We created software that automatically records the times when several significant events happen in a Lisp development session:

- Times when the session is started and stopped.
- Times when different application specific files are opened, closed, and made active in the foreground.
- Times when the two main testing functions were called.

These records provide a less detailed view of the development process. The granularity of the examination is limited to that of system component, without allowing a distinction between different activities performed on each component, like testing, examining, modifying, adding, etc. On the other hand the method addresses the two problems of the audio/video recording method. First it does not interfere with the development process, since it is performed automatically, and second it does not need a time-consuming subsequent coding phase.

Based on these recordings we could then compute the time spent on developing various parts of the system. The entire development process took 28 days. The time spent in Lisp sessions during this interval are shown in Table 8-5 below.

Category			Time in hours	Proportion of total time
Total			67.6	1.00
	Build		5.7	0.08
	Test total		29.8	0.44
		Unassigned	6.1	0.09
		Parse	16.9	0.25
		Trace	6.8	0.10
	Syntax total		15.7	0.23
		General lexicon	1.0	0.02
		Domain lexicon	4.3	0.06
		Grammar	10.4	0.15
	Semantics total		16.4	0.24
		General ontology	0.5	0.01
		Domain ontology	6.6	0.10
		Classification taxonomy	9.4	0.14

Table 8-5. Time spent on Algebra application development

The total time spent working in Lisp on the Algebra application was 67.6 hours. Out of this about half (52%) was spent on building and testing the application, and about half (47%) on developing the syntactic and semantic components. Out of the testing time, about 57% was spent parsing test sentences and examining the results. The rest was split evenly between tracing, a more advanced form of testing where successive steps in the parse process are examined closely, and various other activities that could not be attributed directly to one of the other two.

The other half of the development time, which was consumed on working on individual system components, was split evenly between the syntactic and semantic components. Among the syntactic components, most of the time (66%) was spend on the grammar, and the rest mostly on the domain lexicon. Only a small part (6%) was spent on the domain independent lexicon. The distribution is somewhat similar among the semantic components, with 57% spent on developing the classification taxonomy, 3% on the domain independent knowledge base, and the rest on the Algebra knowledge base.

Comparing the proportion of time spent on various components with the proportion of items added to those components, one can observe an important discrepancy: even if the grammar has the highest reusability rate, about 96-97%, and the lowest renewal rate, about 15-16%, working on it still took about 1/3 of the total time spent on component development (15.7 + 16.4 hours). The explanation comes from the fact that the grammar is a complex of over 2000 unification equations associated with over 150 rules, whose purpose is to be highly selective in allowing only the necessary syntactic patterns, while eliminating as much ambiguity as possible. Thus, most of the time spent on the grammar was not dedicated to adding new rules that address new syntactic constructions. It was rather spent in examining and modifying existing rules so that modified patterns are accepted, without generating a big increase in syntactic ambiguity.

8.5. Conclusions of the development of the Algebra application

The development process of the new Algebra application demonstrated one of the main claims of the advantages brought by our approach: that is a highly reusable approach, which allows developing applications in new semantic domains with relatively small effort.

The total development effort took *less than a month*. The separation of syntax and semantics in the system components allowed us to achieve a high reusability degree of 97% of the syntactic components, and necessitated only about 15% further development. At the same time, the separation of the semantic components between domain independent and domain specific allowed to further reuse about 2/3 of the knowledge base elements. Only the classification taxonomy had to be rebuilt completely from scratch, which was to be expected given the completely different kind of explanations needed in the Algebra application.

One limitation of the argument comes from the relatively smaller size of the knowledge base for Algebra compared to Geometry. This difference indicates that the new domain is relatively simpler, which contributed to some extend to the small development time. Judging by the number of elements it seems to be only about half as complex as the Geometry application. Assuming linearity of the development effort with the number of knowledge base elements, this number means that the development of a similarly complex application would not take more than two months, still a very good time.

One weakness that the study revealed was that even if the grammar had a high reuse rate, further development changes took a disproportionate amount of time with the size of these changes, about 1/3 of total development time. The cause of this mismatch comes from practical requirements imposed on the system. Because the Loom logic system is very slow, a lot of ambiguity that could be solved through semantic selection was instead solved at grammar level. This choice had the negative effect of making the grammar highly complex and brittle when faced with new syntactic patterns. Even so, the total time spent on the grammar was not prohibitively high, only about 10 ½ hours. And the use of a more performing logic system could alleviate the problem considerably, while at the same time improving the robustness of the system.

Chapter 9 Implementing the System in Description Logic

The Loom logic system that constitutes the underlying reasoning mechanism in our approach is one of the many Description Logic systems currently available (for an overview see Möller & Haarslev (2003). In this chapter we discuss what Loom features are used in the implementation of our approach and what are the equivalent Description Logic constructs. Then we examine two alternative logic systems to see if they have the necessary features or not, and/or what changes are necessary to our approach so that it could be implemented in those systems.

The reasons for this effort are:

1. To provide Description Logic system developers with guidance on features needed for NLU in the real world application of tutoring.
2. To help identify alternative logic system candidates for implementing the NLU system, based on their properties.
3. To facilitate the portability of our approach to a new logic system, by identifying specific features used in Loom and the corresponding equivalent constructs in the new system.

9.1. Description Logic features needed for knowledge base modeling

Some of the features of the underlying Loom system are crucial for our approach to building semantic representations. Many of them are widespread among Description Logic systems. Others are not so common, or are found in forms that are not quite adequate for this kind of use. This section identifies the Description Logic features used in the implementation of our approach. We followed the definitions in Baader & Nutt (2003).

In the following sections examples are presented first in Loom format and second in Description Logic format.

9.1.1. Concepts

```
(132) (defconcept Thing
        :implies (:all count Count-Value))
```

DL: Thing $\Rightarrow \forall count.$CountValue

```
(133) (defconcept Element
        :is-primitive Thing)
```

DL: Element ⊑ Thing

```
(134) (defconcept Geometry-Object
        :is-primitive (:and Spatial Named-Object))
```

DL: GeometryObject ⊑ Spatial ⊓ NamedObject

```
(135) (defconcept Congruent-Angles-Reason
        :is (:and Reason
                  (:some topic (:and Angle Congruent))))
```

DL: CongruentAnglesReason ≡ Reason ⊓ ∃*topic*.(Angle ⊓ Congruent)

Example (135) defines concept `Congruent-Angles-Reason` (one of the classification categories used in the Geometry application) as a subconcept of a more general concept `Reason`, with the additional property that it has a `topic` relation whose value satisfies both `Angle` and `Congruent` concepts.

Concept conjunctions (`:and` construct) are introduced in section 9.1.3 below. Concept equalities (`:is`), concept inclusions (`:is-primitive`), and trigger rules (the `:implies` construct in example (132) above) are discussed in sections 9.1.5, 9.1.6, and 9.1.7, while universal (`:all`) and existential (`:some`) role restrictions are presented in sections and 9.1.9 and 9.1.10 respectively.

Concepts form the base of the domain modeling in our approach. They are used to model the main entities of the domain of discourse, as in examples (132), (133), and (134), as well as the classes for the classification task, as shown in example (135). Concepts correspond to unary predicates in First Order Logic. In our application all concepts are derived from a single concept called `Thing`, which acts as a universal concept.

9.1.2. Concept negations

```
(136) (defconcept Non-Conscious-Thing
        :is-primitive (:and Simple-Thing
                            (:not Conscious-Being)))
```

DL: NonConsciousThing ⊑ SimpleThing ⊓ ¬ConsciousBeing

```
(137) (defconcept Infinite-Object
        :is-primitive (:and Open-Object
                            (:not Finite-Object)))
```

DL: InfiniteObject ⊑ OpenObject ⊓ ¬FiniteObject

Example (137) partially defines (uses `:is-primitive` instead of `:is`) `Infinite-Object` as an `Open-Object` that is not a `Finite-Object`.

We used negation mostly to express mutual exclusion between concepts, as the previous examples illustrate. Mutual exclusion is necessary in the determination of unfitness conditions between language elements. Negation is used exclusively on atomic concepts, as the examples above show.

Most of the cases where negation was needed in concept definitions were actually cases of concept covering, when a set of subconcepts fully covers a superconcept (their union logically implies the superconcept). The two examples above fall in this category. Such cases could be represented in Loom by using the mechanism of partitions. Partitions take care of both the covering and the disjointness among its elements. Concept covering was not used because of implementation problems in working in conjunction with instance unification. Partitions could be modeled in Description Logic by using concept disjunction plus negation.

9.1.3. Concept conjunctions

```
(138) (defconcept Measure-Value
        :is-primitive (:and Abstraction Non-Decomposable-Object))
```

DL: $\mathsf{MeasureValue} \sqsubseteq \mathsf{Abstraction} \sqcap \mathsf{NonDecomposableObject}$

The example partially defines the `Measure-Value` concept as a conjunction of two other concepts, `Abstraction` and `Non-Decomposable-Object`.

Concept conjunction is a basic construct used in concept forming expressions everywhere.

9.1.4. Concept disjunctions

```
(139) (defconcept Location-Range
        :is (:or Configuration Decomposable-Object Set))
```

DL: $\mathsf{LocationRange} \equiv \mathsf{Configuration} \sqcup \mathsf{DecomposableObject} \sqcup \mathsf{Set}$

```
(140) (defconcept Spatial-Temporal-Or-Set
        :is (:and Non-Conscious-Thing
                  (:or Spatial-Temporal
                       (:and Set (:all element-role
                                       Spatial-Temporal)))))
```

DL: $\mathsf{SpatialTemporalOrSet} \equiv \mathsf{NonConsciousThing} \sqcap (\mathsf{SpatialTemporal} \sqcup (\mathsf{Set} \sqcap \forall \mathit{elementRole}.\mathsf{SpatialTemporal}))$

```
(141) (implies (:and Non-Conscious-Thing
                     (:or Geometry-Object Measure-Value)
                     (:some location Set)
                     (:some belongs-to (:not Set)))
               (:relates location belongs-to location))
```

DL: $(\mathsf{NonConsciousThing} \sqcap (\mathsf{GeometryObject} \sqcup \mathsf{MeasureValue}) \sqcap \exists \mathit{location}.\mathsf{Set} \sqcap \exists \mathit{belongsTo}.\neg\mathsf{Set}) \Rightarrow \forall \mathit{belongsTo}.\exists \mathit{location} \sqcap \mathit{belongsTo} \circ \mathit{location} \doteq \mathit{location}$

Example (140) defines concept `Spatial-Temporal-Or-Set` as a conjunction between `Non-Conscious-Thing` and a disjunction of two other concept constructs. One is `Spatial-Temporal`, while the other is a `Set` whose `element-role` relations belong to concept `Spatial-Temporal`.

The `:relates` construct is introduced in section 9.1.24. Role-value maps (the $\doteq$ construct) and role composition (the $\circ$ construct) used in the Description Logic translation of example (141) above) are discussed in section 9.1.23.

Concept disjunction is used occasionally in concept definitions or directly in trigger rules to create the right preconditions necessary for inference triggering. The typical case is that when the filler of a role can be either a specific type or a set of elements of that type, like in example (140) above.

9.1.5. Concept equalities

```
(142) (defconcept Collective-Concept
        :is (:and Thing (:the count Collective)))
```

DL: CollectiveConcept ≡ Thing ⊓= 1*count* ⊓ ∃*count*.Collective

```
(143) (defconcept Part
        :is (:and Spatial-Temporal
                  (:some part-of Spatial-Temporal)))
```

DL: Part ≡ SpatialTemporal ⊓ ∃*partOf*.SpatialTemporal

```
(144) (defconcept Closed-Object
        :is (:and Finite-Object (:not Open-Object)))
```

DL: ClosedObject ≡ FiniteObject ⊓ ¬OpenObject

Example (143) defines concept `Part` as a `Spatial-Temporal` concept that has a `part-of` relation with an object of the same `Spatial-Temporal` concept.

Functional role restrictions (the `:the` construct in example (142)) are introduced in section 9.1.11.

The only form of concept equalities accepted in Loom and used in the system are concept definitions, where the left-hand side is an atomic concept. Concept definitions are used to introduce new terminological concepts that are fully defined in terms of other concept expressions. Some of these concepts correspond directly to natural language lexicon entries, like in example (143) above, while others serve other purposes in the inference process. For instance, the `collective-concept` definition in example (142) is used to distinguish plural nouns from singulars.

Concept definitions, together with concept specializations below, organize entities in the domain of discourse in a taxonomy, according to their respective properties. Since Loom classifies only concepts that are not part of a definitional cycle, special care was taken to ensure that the entire taxonomy is acyclic.

Subsumption relations between concepts in the taxonomy are usually expressed explicitly in the definitions, rather than left to be inferred by Loom. Because of this characteristic, in principle full definitions are not necessary; concept specializations could be enough to introduce all necessary terminology. However, the inference processes needed in determining equivalence between different expressions of the same meaning content require that specific Loom instances be classified under trigger concepts. Those trigger concepts need full definitions for the classification process to work.

One important exception in the system, where subsumption relations are completely determined by Loom through concept classification, is that of classification categories used in classifying students' explanations. Since particular semantic configurations have to be classified by Loom into one of the categories, they are all given full concept definitions. As it can be seen below, each of the definitions describes solely the configuration that it needs to identify, with no mention of other categories. The subsumption relations determined by Loom among these categories are used to eliminate more general classifications and keep only the most specific ones.

```
(145) (defconcept Angles-180-Reason
        :is (:and Reason
                  (:some topic (:and Measure-Value
                                     (:all value 180)
                                     (:some measure-of Angle)))))
```

DL: Angles180Reason
≡ Reason ⊓ ∃*topic*.(MeasureValue ⊓ ∀*value*.180 ⊓ ∃*measureOf*.Angle)

```
(146) (defconcept Adjacent-Angles-180-Reason
        :is (:and Reason
                  (:some topic
                         (:and Measure-Value
                               (:all value 180)
                               (:some measure-of
                                      Adjacent-Angle)))))
```

DL: Angles180Reason
≡ Reason ⊓ ∃*topic*.(MeasureValue ⊓ ∀*value*.180 ⊓ ∃*measureOf*.AdjacentAngle)

```
(147) (defconcept Linear-Angles-180-Reason
        :is (:and Reason
                  (:some topic
                         (:and Measure-Value
                               (:all value 180)
                               (:some measure-of
                                      Linear-Angle)))))
```

DL: Angles180Reason
≡ Reason ⊓ ∃*topic*.(MeasureValue ⊓ ∀*value*.180 ⊓ ∃*measureOf*.LinearAngle)

For instance, concept `Linear-Angle` used in the last example above is defined as a subconcept of `Adjacent-Angle` used in the previous example, which is in turn defined as subconcept of `Angle` used in the first example. Thus the first category, `Angles-180-Reason`, subsumes (is more general than) the second one, `Adjacent-Angles-180-Reason`, which in turn subsumes the third one, `Linear-Angles-180-Reason`. When an explanation classifies as all 3 categories, only the last one, which is the most specific one, is kept.

9.1.6. *Concept inclusions*

```
(148) (defconcept Sequence
        :is-primitive Thing)
```

DL: Sequence $\sqsubseteq$ Thing

```
(149) (defconcept Open-Object
        :is-primitive Object)
```

DL: OpenObject $\sqsubseteq$ Object

```
(150) (defconcept Polygon
        :is-primitive (:and Two-D-Object Closed-Object))
```

DL: Polygon $\sqsubseteq$ TwoDObject $\sqcap$ ClosedObject

The last example above partially defines a `Polygon` to be a conjunction of `Two-D-Object` and `Closed-Object`. The partial definition means that the newly introduced concept logically implies its definition, but the reciprocal is not true.

The only form of concept inclusions allowed by Loom is in the form of partial concept definitions or concept specializations, when the left-hand side is an atomic concept. This form is in widespread use all over the knowledge base, because it allows us to avoid defining aspects of concepts that are not relevant for the inference process. Having to fully define all concepts in the knowledge base would make the development process much more tedious, and will also slow down the inference process.

The same effect could be obtained by providing full definitions using extra "difference" concepts. For instance the `open-object` concept in example (149) could be defined as:

```
(151) (defconcept Open-Object
        :is (:and Open Object))
```

DL: OpenObject $\equiv$ Open $\sqcap$ Object

However, in many cases it is hard to even find a good name for the "difference" concept. Concept specializations help avoid cluttering the knowledge base when such concepts are irrelevant for the inference process.

9.1.7. *Trigger rules*

```
(152) (implies Segment (:all equal-to Segment))
```

DL: Segment $\Rightarrow \forall$*equalTo*.Segment

```
(153) (implies Angle (:all equal-to Angle))
```

DL: Angle $\Rightarrow \forall$*equalTo*.Angle

```
(154) (implies Triangle (:all equal-to Triangle))
```

DL: Triangle $\Rightarrow \forall$*equalTo*.Triangle

Example (154) above expresses the logical implication that whenever an object is a `Triangle`, any object linked to it through an `equal-to` relation is also a `Triangle`.

There are cases when the linguistic inference process requires expressing logical implications between already defined concepts that cannot be asserted in the definitions. For instance examples (152), (153), and (154) show a case where one needs to infer that any object that is in an `equal-to` relation to a `Segment`, `Angle`, or `Triangle`, is also a `Segment`, `Angle`, or `Triangle`, respectively. There are various reasons why the implications cannot be stated in the concept definitions. For instance, concept `Segment` is defined as:

```
(155) (defconcept Segment
        :is (:and Linear-Object Open-Finite-Object))
```

DL: Segment ≡ LinearObject ⊓ OpenFiniteObject

If the definition is changed by adding the extra condition in (152):

```
(156) (defconcept Segment
        :is (:and Linear-Object Open-Finite-Object
                  (:all equal-to Segment)))
```

DL: Segment ≡ LinearObject ⊓ OpenFiniteObject ⊓ ∀*equalTo*.Segment

then the definition becomes circular (the definition of concept `Segment` uses `Segment` inside the definition) and Loom cannot classify it anymore. Even if circularity were not a problem, we still have another difficulty: the definition states an equivalence relation between the concept name and the concept description. Adding conditions to the concept description would add extra preconditions to the inverse (right to left) implication, making the inference process potentially fail. For instance in the case above we might have an object that is known to be `Linear-Object` and `Open-Finite-Object`, but its is not known whether all objects it is in an `equal-to` relation to are of type `Segment`. Then, with the modified definition, that object would not be classified as a `Segment`, even if it should.

For such cases we used Loom's `implies` construct. Its equivalent in the Description Logic language is the trigger rule. According to Baader & Nutt (2003) the semantics of the trigger rule differs from that of the inclusion by the fact that the contrapositive is not taken to also be true for the trigger rule, as in the case of concept inclusion. That is, if we have rule $A \Rightarrow B$, this is not equivalent to $\neg B \Rightarrow \neg A$, while $A \sqsubseteq B$ is indeed equivalent to $\neg B \sqsubseteq \neg A$. An analysis of all cases where we use the `implies` construct shows that the distinction is not useful in our application. As it can be seen in examples (152), (153), and (154), the contrapositive also holds. Actually in these cases even the reciprocal holds, so the more adequate logical expression would be for the first case:

(157) DL: Segment ≡ ∀*equalTo*.Segment

However this would constitute a second definition for concept `Segment`, and Loom, as most other Description Logic languages, require concept definitions to be unique. Even when the reciprocal does not hold, the contrapositive does. But even an expression like:

(158) DL: Segment ⊑ ∀*equalTo*.Segment

cannot be stated in Loom, since the specialization is still considered a second definition.

9.1.8. *Roles*

```
(159) (defconcept Single-Thing
        :is (:and Thing (:some count Single)))
```

DL: SingleThing ≡ Thing ⊓ ∃*count*.Single

The example defines a `Single-Thing` to be a `Thing` that has a `count` relation whose value belongs to the `Single` concept.

Roles or relations are a basic mechanism of any Description Logic language. They are in widespread use in the knowledge base, modeling relationships between the entities in the domain of discourse. As in most Description Logic languages, Loom only accepts binary roles, corresponding to binary First Order Logic predicates.

9.1.9. *Universal role restrictions/quantifications*

```
(160) (defconcept Pair-Of-Numbers
        :is (:and Pair
                  (:all element-role Natural-Number)))
```

DL: PairOfNumbers ≡ Pair ⊓ ∀*elementRole*.NaturalNumber

```
(161) (defconcept Provide
        :is-primitive Directed-Action
        :implies (:all recipient Person))
```

DL: Provide ⊑ DirectedAction
Provide ⊑ ∀*recipient*.Person

The first example defines the `Pair-Of-Numbers` concept to be a `Pair` concept whose `element-role` relations all belong to the `Natural-Number` concept.

Universal role quantifications are only occasionally used in full concept definitions or equalities. The main reason for its avoidance being that in an open-world reasoning system as Loom it is usually difficult to prove that the universal quantification holds, and then the right-to-left implication in a concept definition is not very useful. Universal role quantifications are however in widespread use in concept specializations (partial definitions).

9.1.10. *Full existential role restrictions/quantifications*

```
(162) (defconcept Quantity
        :is (:and Measure-Value (:some value Number)))
```

DL: Quantity ≡ MeasureValue ⊓ ∃*value*.Number

```
(163) (defconcept Part
        :is (:and Spatial-Temporal
                  (:some part-of Spatial-Temporal)))
```

DL: Part ≡ SpatialTemporal ⊓ ∃*partOf*.SpatialTemporal

Example (162) defines concept `Quantity` as a `Measure-Value` concept whose `value` attribute is a `Number`.

Existential role quantification is in widespread use in concept definitions and specializations. All existential quantifiers are qualified with a type, even if sometimes it is the universal type `Thing`.

9.1.11. Functional role restrictions

```
(164) (defconcept Collective-Thing
        :is (:and Thing
                  (:the count Collective)))
```

DL: CollectiveThing ≡ Thing ⊓= 1*count* ⊓ ∃*count*.Collective

```
(165) (defconcept Pair
        :is (:and Set (:the cardinality 2)))
```

DL: Pair ≡ Set ⊓= 1*cardinality* ⊓ ∃*cardinality*.2

The last example defines the `Pair` concept to be a `Set` whose unique `cardinality` relation has the value `2`.

Concrete domains, like natural numbers used in example (165), are introduced in section 9.1.25 below.

Functional roles are used all over the knowledge base to ensure uniqueness of roles when applied to specific concepts. The =1 Description Logic operator for which Loom provides the special construct `:the` can be defined in terms of unqualified number restrictions by the relation:

(166) DL: $=\mathrm{nr} \equiv \geqslant \mathrm{nr} \sqcap \leqslant \mathrm{nr}$

Thus its use on a role in a concept definition means that the role represents a total function when applied to the defined concept. Partial functions have also been used using only an upper bound on the role fillers:

```
(167) (defconcept Subtract
        :is-primitive (:and Being&Having
                            (:all component Measure-Value)
                            (:at-most 1 sum)))
```

DL: Subtract ⊑ Being & Having ⊓ ∀*component*.MeasureValue ⊓ ⩽ 1*sum*

Functional roles can also be expressed more generally at the level of the role definitions, in which case the restriction has effect for all uses of the role. Both forms are used in the knowledge base:

```
(168) (defrelation count
        :is-primitive (:and relation
                            (:domain Thing)
                            (:range Count-Value))
        :characteristics :single-valued)
```

DL: *count* ⊑ *relation*
∃*count*. ⊤ ⊑ Thing ⊓ ∀*count*.CountValue ⊓ ⩽ 1*count*

This example partially defines relation `count` by specifying the type of its domain and range, and also declaring it to be `:single-valued`, which Loom's way to say it is a functional role.

Domain and range restrictions are discussed in section 9.1.22.

9.1.12. Unqualified number restrictions

```
(169) (defconcept Add-To-Configuration
        :is (:and Add-To
                  (:at-least 1 component)))
```

DL: AddToConfiguration ≡ AddTo ⊓ ⩾ 1*component*

```
(170) (defconcept Subtract
        :is-primitive (:and Being&Having
                            (:all component Measure-Value)
                            (:at-most 1 sum)))
```

DL: Subtract ⊑ Being & Having ⊓ ∀*component*.MeasureValue ⊓ ⩽ 1*sum*

Example (170) above gives a partial definition (one-way implication) for the configuration concept `Subtract` as a subconcept of the more general configuration `Being&Having`, whose `component` arguments all belong to the `Measure-Value` concept, and which has at most one `sum` role.

The lower bound number restriction with a limit of 1 is used to specify existential roles with no explicit role restriction, as in example (169). It can be substituted with the existential role with a universal concept restriction:

(171) DL: ⩾ 1*r* ≡ ∃*r*.⊤

Such a construct was particularly useful in defining "relationship roles" (Borgida & Brachman, 2003). These are concepts that are defined by the fact that they participate in a specific relationship, like the example below:

```
(172) (defconcept Equal-Thing
        :is (:at-least 1 equal-to))
```

DL: EqualThing ≡ ⩾ 1*equalTo*

The upper bound number restriction with a limit of 1 is used to specify partial functional roles with no explicit role restriction:

```
(173) (defconcept Measure-Value
        :is-primitive (:and Abstraction Nondecomposable-Object
                            (:at-most 1 value)
                            (:at-most 1 unit)))
```

DL: MeasureValue ⊑ Abstraction ⊓ NonDecomposableObject ⊓ ⩽ 1*value* ⊓ ⩽ 1*unit*

Number restrictions with a limit other than 1 have been used in the model in only one case: to identify pairs as sets with 2 elements, using the following implication in combination with the definition of `Pair` given in example (165):

```
(174) (implies (:and Set (:exactly 2 element-role))
               (:the cardinality 2))
```

DL: $\mathsf{Set} \sqcap = 2\mathit{elementRole} \Rightarrow = 1\mathit{cardinality} \sqcap \forall \mathit{cardinality}.2$

Qualified number restrictions have not been found necessary in our application. Unqualified number restrictions have been used only on atomic roles.

9.1.13. Role hierarchies/inclusions

```
(175) (defrelation belongs-to
        :is-primitive relation)
```

DL: $\mathit{belongsTo} \sqsubseteq \mathit{relation}$

```
(176) (defrelation part-of
        :is-primitive belongs-to)
```

DL: $\mathit{partOf} \sqsubseteq \mathit{belongsTo}$

The last example above provides a partial definition of relation `part-of`, by specifying it is a more specific relation to `belongs-to`.

Role hierarchies are in widespread use in the knowledge base to model relational words of the natural language. One such case is that of generic prepositions, which start as very general roles, and can be further specialized when applied to specific concepts. In Description Logic role hierarchies can be expressed by role inclusion axioms, as seen above.

9.1.14. Role definitions/equalities

```
(177) (defrelation has
        :is (:inverse belongs-to))
```

DL: $\mathit{has} \equiv \mathit{belongsTo}^{-}$

```
(178) (defrelation has-part
        :is (:inverse part-of))
```

DL: $\mathit{hasPart} \equiv \mathit{partOf}^{-}$

```
(179) (defrelation measure-of
        :is (:and belongs-to
                  (:domain Measure-Value)
                  (:range Spatial-Temporal)))
```

DL:
$\mathit{measureOf} \sqsubseteq \mathit{belongsTo}$
$\exists \mathit{measureOf}.\top \sqsubseteq \mathsf{MeasureValue} \sqcap \forall \mathit{measureOf}.\mathsf{SpatialTemporal}$
$\mathsf{MeasureValue} \sqcap \exists \mathit{belongsTo}.\mathsf{SpatialTemporal}$
$\sqsubseteq \exists \mathit{measureOf}.\top \sqcap \mathit{measureOf} \doteq \mathit{belongsTo}$

Example (179) above defines relation `measure-of` as a specialization of the more general relation of relation `belongs-to`, whose domain is a `Measure-Value` and whose range is a `Spatial-Temporal` concept.

Role equalities were needed in our model mostly to give names to inverse relations (see section 9.1.21 below). There were a few cases however where role equalities were needed in relation definitions, as seen in example (179). Role equalities are of course equivalent to a pair of role inclusions between the two sides. Thus role equalities were used in cases when we needed the system to use the reciprocal implication in a forward-chaining inference process, to infer a more specialized role when all conditions in its definition are present.

9.1.15. General role axioms

```
(180) (implies (:and possessed
                     (:domain Generalized-Possession)
                     (:range Measure-Value))
               attribute)
```

DL: $\text{GeneralizedPossession} \sqcap \exists \textit{possessed}.\text{MeasureValue} \Rightarrow \exists \textit{attribute}.\top \sqcap \textit{attribute} \doteq \textit{possessed}$

The role implication above expresses the fact that a `possessed` relation with a domain of a `Generalized-Possession` and a range of a `Measure-Value` concept is also an `attribute` relation.

General role axioms were needed in the domain model to express implications between different relations under specific conditions. In some cases, when the consequent is an atomic role, the same implications could have been modeled through role definitions, except that Loom has the same restriction for roles as for concepts of only allowing single definitions.

The semantics of the `implies` construct for relations is the same as the one for trigger rules for concepts, in that it does not imply the fact that the contrapositive implication is also valid. Thus it can be modeled in Description Logic through a trigger rule for relations. However, as for concepts, the distinction was not useful in our application, all such cases could be modeled with role implications.

The example above had to be modeled in Description Logic through concept implications instead, because of the presence of domain and range restrictions, which do not have a direct corresponding construct in the role language in Description Logic (see section 9.1.22 below).

9.1.16. Role conjunctions

```
(181) (defrelation count
        :is-primitive (:and relation
                            (:domain Thing)
                            (:range Count-Value)))
```

DL: $\textit{count} \sqsubseteq \textit{relation}$
$\exists \textit{count}.\top \sqsubseteq \text{Thing} \sqcap \forall \textit{count}.\text{CountValue}$

This example provides a partial definition for relation `count`, as a specialization of the generic role `relation`, with a `Thing` domain and a `Count-Value` range.

Conjunction on roles has been used to specify domain and range restrictions on newly defined relations in role hierarchies. The domain and range restrictions could be modeled in Description Logic with corresponding conditions on the related concepts, as seen in the example above. However being able to specify the restrictions as part of a relation definitions makes the role hierarchy conceptually clearer.

9.1.17. Role disjunctions

Role disjunction is not allowed by Loom, so it has not been used in our application. Cases where disjunction might have been useful have been modeled using the equivalent conjunction form instead.

9.1.18. Role complements

Role complement has not been used in our application either, as it is also not part of the Loom definitional language. The construct was not needed in modeling Geometry.

9.1.19. Role transitivity

```
(182) (implies (:compose belongs-to belongs-to)
               belongs-to)
```

DL: $belongsTo \circ belongsTo \Rightarrow belongsTo$

This example expresses transitivity of relation `belongs-to` by specifying that a chain of two such relations implies the relation being present between the ends of the chain.

Some of the relations used in the knowledge base do have the transitivity property. Since transitivity is not supported in Loom directly, it was modeled through a combination of role implication and role composition. However, some Description Logic systems support role transitivity but not role composition, so it was listed as a separate feature.

9.1.20. Role compositions

```
(183) (implies (:compose belongs-to belongs-to)
               belongs-to)
```

DL: $belongsTo \circ belongsTo \Rightarrow belongsTo$

```
(184) (implies (:and Ascription
                     (:some attribute (:some equal-to Thing)))
               (:and Equals
                     (:some goal Thing)
                     (:same-as goal
                               (:compose attribute equal-to))))
```

DL: $\text{Ascription} \sqcap \exists attribute.\exists equalTo.\text{Thing} \Rightarrow \exists goal.\text{Thing} \sqcap goal \doteq attribute \circ equalTo$

The implication in the last example above expresses the fact that an `Ascription` configuration with an `attribute` functional role that has an `equal-to` relation to some object of the generic type `Thing` can also be seen as an `Equals` configuration with a

`goal` functional role which is the object that is the target of the original `equal-to` relation.

Role composition is used in the domain model to express equivalence between relations of different kinds or specific properties of relations. Only composition chains with a length of 2 were needed in the knowledge base. In example (183) above, role composition is used to express transitivity.

9.1.21. Inverse roles

```
(185) (defrelation has
        :is (:inverse belongs-to))
```

DL: $has \equiv belongsTo^{-}$

```
(186) (defrelation has-part
        :is (:inverse part-of))
```

DL: $hasPart \equiv partOf^{-}$

Inverse roles have been used extensively to ensure relations between instances composing the semantic representations can be navigated in both directions. This makes the definition of classification categories much easier and reduces the number of classes. Inverse roles were applied in the system only on atomic roles.

One particular category of inverse roles used in our implementation is symmetric roles. Those are roles whose inverse is identical to the direct role. Since Loom prohibits the use of circular relation definitions, they cannot be declared using the `:inverse` construct. Loom provides a special declaration for symmetric roles:

```
(187) (defrelation opposite-to
        :is-primitive (:and relation
                            (:domain Open-Finite-Object)
                            (:range Open-Finite-Object))
        :characteristics (:single-valued :symmetric))
```

$oppositeTo \sqsubseteq relation$

DL: $oppositeTo \equiv oppositeTo^{-}$

$\exists oppositeTo.\top \sqsubseteq \text{OpenFiniteObject} \sqcap \forall oppositeTo.\text{OpenFiniteObject}$

This example gives a partial definition of the relation `opposite-to` as a specialization of the generic role `relation` whose domain and range are of type `Open-Finite-Object`. At the same time the defined relation is declared to by a functional role (the `:single-valued` declaration) and symmetric.

9.1.22. Domain and range role restrictions

```
(188) (defrelation congruent-to
        :is (:and equal-to
                  (:domain Finite-Object)
                  (:range Finite-Object)))
```

congruentTo ⊑ *equalTo*

DL: ∃*congruentTo*.⊤⊑ FiniteObject ⊓ ∀*congruentTo*.FiniteObject
FiniteObject ⊓ ∃*equalTo*.FiniteObject ⊑ ∃*congruentTo*.⊤ ⊓ *congruentTo* ≐ *equalTo*

```
(189) (defrelation intersects
        :is-primitive (:and relation
                            (:domain Geometry-Object)
                            (:range Geometry-Object)))
```

DL: *intersects* ⊑ *relation*
∃*intersects*.⊤⊑ GeometryObject ⊓ ∀*intersects*.GeometryObject

Example (188) defines relation `congruent-to` as being logically equivalent to the more general relation `equal-to`, when its domain and range are of type `Finite-Object`.

Domain and range restrictions are constructs specific to Loom, which were used extensively in defining new relations in the knowledge base. These constructs allowed us to specify logical restrictions inherent to the meaning of a relation in its definition, rather than having to specify them in all places where the relation is used.

There is no direct equivalent for domain and range restrictions in Description Logic. They can be modeled equivalently through concept implications, as seen in the examples above. The equivalent Description Logic expressions given above might not be completely equivalent from the inference services point of view, because they are not part of the relation definition. It is not clear if the logic system can use these implications in determining subsumption relationship among relations. For instance, in example (190) below, as in example (188), the reciprocal implication needs to use a role-value map (see section 9.1.23 below) whose semantics is weaker than that of a role implication.

```
(190) (defrelation measure-of
        :is (:and belongs-to
                  (:domain Measure-Value)
                  (:range Spatial-Temporal)))
```

measureOf ⊑ *belongsTo*

DL: ∃*measureOf*.⊤ ⊑ MeasureValue ⊓ ∀*measureOf*.SpatialTemporal
MeasureValue ⊓ ∃*belongsTo*.SpatialTemporal
⊑ ∃*measureOf*.⊤ ⊓ *measureOf* ≐ *belongsTo*

9.1.23. Role-value maps (agreements)

```
(191) (implies (:and Get (:some actee Element)
                         (:some property Thing))
               (:same-as actee property))
```

DL: Get ⊓ ∃*actee*.Element ⊓ ∃*property*.Thing ⇒ *actee* ≐ *property*

```
(192) (implies (:and Generalized-Possession
                     (:some part Collective-Thing)
                     (:some whole Set-Of-Collectives))
               (:same-as part (:compose whole element-role)))
```

DL: $\mathsf{GeneralizedPossession} \sqcap \exists \mathit{part}.\mathsf{CollectiveThing} \sqcap \exists \mathit{whole}.\mathsf{SetOfCollectives} \Rightarrow \mathit{part} \doteq \mathit{whole} \circ \mathit{elementRole}$

The first example above expresses the logical implication that whenever a configuration of type `Get` has an `actee` functional role which is an `Element` and a `property` role of the generic type `Thing`, the two must be filled by the same object.

Role-value maps specify equality restrictions on fillers of different roles of the same concept. Equality role-value maps were used in our knowledge base to model cases where two different roles are supposed to be filled by the same individual. One important aspect of our use is that Loom restricts role-value maps to functional roles, so only such cases were used in the system.

Another aspect worth mentioning is that because the skolem instances used in the system do not follow the unique name assumption (see section 9.2 below), such agreement maps might result in two instances being unified into a single one. This is also made possible by the restriction of role-value maps to functional roles, since for multiple-value roles it would be hard to know what pairs of values to unify.

In most cases equality maps were used directly on atomic roles. There were a few cases, like in example (192), where one of the two arguments had to be a role composition of length 2.

9.1.24. ':relates' constructs

```
(193) (implies (:and Result (:some cause Being&Having)
                            (:some effect Spatial-Temporal))
               (:relates results-in cause effect))
```

DL: $\mathsf{Result} \sqcap \exists \mathit{cause}.\mathsf{Being\ \&\ Having} \sqcap \exists \mathit{effect}.\mathsf{SpatialTemporal} \Rightarrow \mathit{cause} \circ \mathit{resultsIn} \doteq \mathit{effect}$

```
(194) (implies (:and Equal-Thing Measure-Value
                     (:some measure-of Open-Object))
               (:relates congruent-to measure-of
                         (:compose equal-to measure-of)))
```

DL: $\mathsf{EqualThing} \sqcap \mathsf{MeasureValue} \sqcap \exists \mathit{measureOf}.\mathsf{OpenObject} \Rightarrow \mathit{measureOf} \circ \mathit{congruentTo} \doteq \mathit{equalTo} \circ \mathit{measureOf}$

The first example above states the implication that whenever a `Result` configuration has a `cause` functional role of type `Being&Having` and an `effect` functional role of type `Spatial-Temporal`, the fillers of the two roles must also be connected through a `results-in` relation.

One construct specific to Loom that was used in various parts of the Knowledge base is the `:relates` construct. This construct specifies the fact that a given relation is valid between the fillers of two existing roles. The roles involved are single-value roles. An

equivalent Description Logic construct could be the equality role-value maps used in conjunction with role composition, as seen in the examples above.

There is however an important distinction between the `:relates` and the `:same-as` constructs in Loom, which is not reflected in the use of the same Description Logic translation for both of them. For the `:same-as` construct, the relations are supposed to be already present, the only added constraint being the equality of the respective role filler objects. For the `:relates` construct, the first relation can be a newly asserted relation, which connects existing objects. Thus, an extra existential role restriction is needed for the filler of one of the two connected roles in order to get a more accurate Description Logic translation:

```
(195) (implies (:and Result (:some cause Being&Having)
                            (:some effect Spatial-Temporal))
               (:relates results-in cause effect))
```

DL: $\mathsf{Result} \sqcap \exists \mathit{cause}.\mathsf{Being} \,\&\, \mathsf{Having} \sqcap \exists \mathit{effect}.\mathsf{SpatialTemporal}$
$\Rightarrow \forall \mathit{cause}.\exists \mathit{resultsIn}.\top \sqcap \mathit{cause} \circ \mathit{resultsIn} \doteq \mathit{effect}$

```
(196) (implies (:and Equal-Thing Measure-Value
                     (:some measure-of Open-Object))
               (:relates congruent-to measure-of
                         (:compose equal-to measure-of)))
```

DL: $\mathsf{EqualThing} \sqcap \mathsf{MeasureValue} \sqcap \exists \mathit{measureOf}.\mathsf{OpenObject}$
$\Rightarrow \forall \mathit{measureOf}.\exists \mathit{congruentTo}.\top$
$\sqcap \mathit{measureOf} \circ \mathit{congruentTo} \doteq \mathit{equalTo} \circ \mathit{measureOf}$

9.1.25. Concrete domains

```
(197) (defrelation cardinality
        :domain Set
        :range Natural-Number
        :characteristics :single-valued)
```

DL: $\mathit{cardinality} \sqsubseteq \mathit{relation}$
$\mathsf{Set} \sqcap \exists \mathit{cardinality}.\top \sqsubseteq \leqslant 1 \mathit{cardinality} \sqcap \forall \mathit{cardinality}.\mathsf{NaturalNumber}$

```
(198) (defconcept Pair
        :is (:and Set (:the cardinality 2)))
```

DL: $\mathsf{Pair} \equiv \mathsf{Set} \sqcap = 1 \mathit{cardinality} \sqcap \exists \mathit{cardinality}.2$

The first example defines relation `cardinality` as a primitive (by default) functional role whose domain is a `Set` and the range is a `Natural-Number`.

The only concrete domain used in our application is that of numbers, both integer and real. It has been used mainly for two purposes. First, to express cardinality of sets (see examples (197) and (198) above).

```
(199) (defrelation value
        :range Number
        :characteristics :single-valued)
```

DL: $value \sqsubseteq relation$
$\exists value.\top \sqsubseteq \leq 1 value \sqcap \forall value.\text{Number}$

```
(200) (defconcept Angles-180-Reason
        :is (:and Reason
                  (:some topic (:and Measure-Value
                                     (:all value 180)
                                     (:some measure-of Angle)))))
```

DL: $\text{Angles180Reason} \equiv \text{Reason} \sqcap \exists topic.(\text{MeasureValue} \sqcap \forall value.180 \sqcap \exists measureOf.\text{Angle})$

And second, to model numerical values in the domain of discourse, like angle measures (examples (199) and (200)). Except for detection of equality, no other predicate was used on numbers in our application.

9.2. Representing semantic structures in Description Logic

As shown in section 5.3.3, semantic structures representing natural language meaning are expressed in our system through Loom instances. The equivalent Description Logic constructs for Loom instances are ABox individuals. Thus the semantic representation of a natural language sentence can be described in Description Logic as a set of ABox individuals and the associated set of assertions on them. The assertions can be of two kinds: concept assertions, which state that an individual belongs to a given concept, and role assertions, which state that a given role holds between two individuals.

For example, let's take again the sentence in example (52) repeated below:

(201) The measure of a right angle is 90 degrees.

And whose semantic representation in Loom is given in example (56) repeated here:

```
(202) (tell (:about measure-1
              (:create Angle-Measure)
              (unit 'degree)
              (value 90)
              (measure-of angle-1)))
      (tell (:about angle-1
              (:create Right-Angle)
              (measure measure-1)))
      (tell (:about being&having-1
              (:create Ascription)
              (attribute measure-1)
              (attribuend measure-1)))
```

The same representation can be expressed in Description Logic as below:

(203) DL: AngleMeasure(MEASURE1)
Unit(MEASURE1,DEGREE)
Value(MEASURE1,90)
MeasureOf(MEASURE1,ANGLE1)

DL: RightAngle(ANGLE1)
Measure(ANGLE1,MEASURE1)

DL: Ascription(BEING & HAVING1)
Attribute(BEING & HAVING1,MEASURE1)
Attribuend(BEING & HAVING1,MEASURE1)

One important difference however between our use of Loom instances and the properties usually assumed for Description Logic individuals is that we used Loom instances that do not work under the unique name assumption (called skolems in Loom). Thus two instances that are created separately, with different names, can in the end be merged, and thus having a single individual correspondent in the final equivalent Description Logic representation.

The process of merging instances can be assimilated to asserting equality relations among them. However under the unique name assumption an equality relation can only check that the two instances are actually the same. If we do not assume individuals with different names are different, asserting equality can lead to a unification process performed on the concept descriptions of the given individuals (taken as the set of all predicates that hold for the given individual). This process can end in one of two results: either an individual with a concept description equivalent to the most general unifier of the two given individuals, if that exists, or an individual with an incoherent (contradictory) description, in case the unification fails.

9.2.1. Linguistic inference in Description Logic

Section 5.3.5 presented the four different methods used in semantic restriction statements attached to grammar rules, which generate the semantic representations of a phrase compositionally from the semantic representation of its elements. These four methods have the following Description Logic equivalents:

1. `create-semantics` in Loom terms simply creates a new instance and asserts it to be of the type corresponding to the word it represents. Thus in Description Logic it corresponds to an ABox assertion of the right type on a newly named individual. As was mentioned earlier, Loom skolem instances do not observe the unique name assumption. So the corresponding Description Logic individuals will not work under the unique names assumption either. For instance, if `Angle` is the concept for `x1` below:

(204) `(create-semantics (x1 concept))`

DL: Angle(ANGLE1)

2. `combine-semantics` takes two Loom instances and merges them into a single one. The result of the operation is that the resulting instance inherits all properties of the

two original instances. In case the two sets of properties are logically incompatible, Loom marks the resulting instance as being incoherent. As shown in the previous section, the corresponding Description Logic operation would be to assert an equality relation on the respective individuals. Thus if `Angle1` and `Adjacent1` are the instances representing the semantics of `x2` and `x1` below:

```
(205) (combine-semantics (x2 semantics) (x1 semantics))
```

DL: ANGLE1=ADJACENT1

3. `connect-semantics` takes two Loom instances, one of which is a predicate and the other an argument, and connects them through a role corresponding to semantic role of the argument for the given predicate. In Description Logic the equivalent is to simply assert the corresponding role on the individuals that model the two Loom instances. For example, in case the two instances are `Being&Having1` and `Angle1`, and the semantic role is `attribuend`:

```
(206) (connect-semantics (x1 semantics)
                         (x1 relation) (x2 semantics))
```

DL: *attribuend*(BEING & HAVING1, ANGLE1)

4. `collect-semantics` takes the two Loom instances and adds them as elements of a special construct concept, called `Sequence`. The classification function then knows to extract all members of such a construct and classify them separately. For instance if the two instances are `Being&Having1` and `Equals1`:

```
(207) (collect-semantics (x1 semantics) (x2 semantics))
```

Sequence(SEQUENCE1)
DL: *sequenceElement*(SEQUENCE1,BEING & HAVING1)
sequenceElement(SEQUENCE1,EQUALS1)

9.2.2. Reference resolution

As described in section 5.3.8, in our approach the resolution of references takes place at the semantic level. The mechanism used is that of merging Loom skolem instances representing the referent and the antecedent, with the corresponding Description Logic of asserting equality over individuals.

However, it is often the case that there is a list of possible antecedents for a given reference, with no way to know in advance which ones to merge. Thus, unlike in the `combine-semantics` operation above, a failure of the unification of the two instances should only result in abandoning the respective resolving pair, not in abandoning the entire parse.

There are two possible mechanisms that allow for this behavior. The first one, which was not used in our system for practical reasons, is that of non-monotonic reasoning. Under this approach, trying a resolution pair would be equivalent to asserting equality between the respective Description Logic individuals. In case the unification process fails, the equality assertion would be retracted and a different one would be tried.

The second mechanism, which was used in our system because it is needed in the normal parsing process anyway, is that of separate contexts. In this approach, the tentative unification is tried in a separate semantic context, represented by a Loom workspace. In case the unification succeeds, the semantic context is retained. If not, it is abandoned. The following section gives a Description Logic equivalent for Loom workspaces.

9.2.3. Semantic contexts in Description Logic

In section 5.3.6 it was shown that the parallel nature of the parser leads to a need to keep separate parallel semantic contexts for the semantic representations of the various parses. The mechanism used in our system for this purpose is that of Loom workspaces. The previous section shows that workspaces are useful for the reference resolution process too.

An equivalent Description Logic mechanism for Loom workspaces would be to have multiple ABoxes. In such a system, all assertions on individuals would be done relative to a single ABox. All ABoxes however need to be able to see all the knowledge base concept and relation inference rules. Besides, an individual ABox needs to see only a subset of individuals, those created on the same path in the parsing process, together with all assertions made on them.

One important requirement resulting from the parsing process is that a new ABox might need to see the individuals from several different ABoxes. For instance, when combining the subject noun phrase and the predicate verb phrase into a clause, the ABox corresponding to the clause needs to have access both to the individuals representing the subject and to those representing the predicate.

This partitioning of the space of individuals among ABoxes could be implemented through two different mechanisms. One is to have the ability to create a new ABox as a copy of an existing ABox. This involves a high cost at creation time, but then all assertions can be made in the new ABox, independent of any other. However, because of reasons shown in the previous paragraph, the new ABox needs to copy instances not just from one, but from several existing ABoxes.

The other, which is used in Loom, is to have the ability to define an inheritance relation among ABoxes (workspaces respectively). Under the inheritance mechanism, an ABox would only see individuals and assertions made in ABoxes it inherits from. No copying of individuals is needed at creation time. However, the inference process has to keep track of what ABox each assertion was made in, and make new assertions visible only in ABoxes down the inheritance hierarchy. Again, for reasons presented above, the system has to allow for multiple inheritance among ABoxes.

9.3. Inference services used in the NLU system

Our system makes use of several different reasoning services provided by the Loom logic system in the process of capturing the meaning of natural language. Similar services are needed in an equivalent Description Logic model. This section identifies the main inference services that were used in the development and run of the system.

One main characteristic of the specific way logic is used in our system is that the knowledge base, which is the definitional part of the model, is static. All concepts are predefined, no concepts are added, deleted or changed during the work of the system. Because of this characteristic, most reasoning services on concepts available in Loom were useful mainly to assist in the development of the system. Inferences performed during the running of the system concerned mainly individuals used in the semantic representation.

9.3.1. Concept subsumption/classification

Concept classification was not used much in modeling the domain of discourse. All desired subsumption relations were declared explicitly. Subsumption was occasionally used during the development process to check the consistency of the model.

The only place where subsumption was used extensively was the classification taxonomy. All classification categories were defined independent of one another, their relative relationship being entirely determined by the Loom system. Subsumption was used mainly by classifying the entire knowledge base once all concepts were defined.

9.3.2. Concept satisfiability and equivalence

Satisfiability and equivalence were used during the development of the domain model to check for consistency of the model. Basically, every time a concept is added or changed, Loom checks it for satisfiability with respect to the rest of the TBox. Similarly, the new concept is checked to see if it is equivalent to an already defined concept.

9.3.3. Concept disjointness

Similarly to previous services, disjointness is used in the development process to make sure concepts that are meant to be incompatible to one another stay that way. However, disjointness plays an important role in the working of the system too. Disjointness between concepts was the basic mechanism used in the system to enforce semantic constraints. Whenever there was a need to make sure two semantic representations cannot be put together in a specific way, the corresponding concepts were defined as (or made to inherit from) disjoint concepts. The mechanism used to define disjointness was concept negation, as it can be seen in examples (136) and (137) in section 9.1.2, repeated below for convenience.

(208)
```
(defconcept Non-Conscious-Thing
  :is-primitive (:and Simple-Thing
                      (:not Conscious-Being)))
```

DL: $\text{NonConsciousThing} \sqsubseteq \text{SimpleThing} \sqcap \neg\text{ConsciousBeing}$

(209)
```
(defconcept Infinite-Object
  :is-primitive (:and Open-Object
                      (:not Finite-Object)))
```

DL: $\text{InfiniteObject} \sqsubseteq \text{OpenObject} \sqcap \neg\text{FiniteObject}$

9.3.4. Individual consistency

During the parse process, each time a new constituent was built, its corresponding semantic structure is checked for semantic validity. This is done by taking the Loom instance that represents the given constituent and checking it for consistency with respect to the knowledge base. If the given instance was asserted to belong to incompatible (disjoint) concepts, Loom marks it as incoherent. In such a case the parser abandons the current parse path.

An instance can become incoherent as the result of the two main actions that are performed by the linguistic inference module: `combine-semantics` and `connect-semantics`. In `combine-semantics` the two existing instances are merged together into a new one. It is possible that the concept descriptions for the two instances to be incompatible, because they inherit from disjoint concepts. In `connect-semantics` an instance is connected through a binary relation to another instance. In this case it is possible that the concept corresponding to one of the two instances puts constraints on the other end of the relation that are incompatible with the actual concept for the given instance. It is also possible that restrictions placed on the domain and range of the relation through `:domain` and `:range` constructs (see section 9.1.22) are incompatible with the properties asserted on the respective instances. All these cases are captured by the instance coherence test.

Another situation where individual consistency was used in the system is reference resolution. As shown in section 5.3.8, solving anaphors against potential antecedents is done in the system by trying to merge the corresponding Loom instances with the `combine-semantics` operator. The result is then checked for consistency, and if the check succeeds, it is kept as a valid resolution. If not, the next potential antecedent is checked, until a semantically compatible one is found or the list is exhausted.

ABox consistency as a whole was not tested in the system because all instances used in the system are created and checked individually at multiple time points during the parse.

9.3.5. Individual realization/classification

An important service for the work of the system is that of instance realization. Whenever a new property or role is asserted on an existing individual as a result of an action performed by the linguistic inference module, it has to be checked to see if the new property leads to a more specific classification. This is important for two different reasons. First, the more specific classification could be the precondition of a conceptual implication, which needs to be applied as the result of the new classification. Second, the same mechanism is used in the classification of the sentence meaning on the explanation taxonomy.

```
(210) (defconcept Equal-Thing
        :is (:at-least 1 equal-to))
```

DL: EqualThing $\equiv\geqslant 1 equalTo$

```
(211) (implies (:and Ascription (:some attribute Equal-Thing))
               Equal-Configuration)
```

DL: $\text{Ascription} \sqcap \exists \textit{attribute}.\text{EqualThing} \Rightarrow \text{EqualConfiguration}$

For example, if we take the definitions above, if we have an individual that is linked to another one through an `equal-to` relation, it needs to be recognized as an `Equal-Thing` concept. This concept is then used as a precondition in an implication that identifies a specific type of configuration.

9.3.6. Role specialization

Similar to the case of individuals, a given role can be further specialized when new properties are asserted on existing individuals. This can happen either when a new role is asserted to link existing individuals, or when a property is asserted on an individual linked by some role to another one.

```
(212) (defrelation measure-of
        :is (:and belongs-to
                  (:domain Measure-Value)
                  (:range Spatial-Temporal)))
```

DL:
$$\textit{measureOf} \sqsubseteq \textit{belongsTo}$$
$$\exists \textit{measureOf}.\top \sqsubseteq \text{MeasureValue} \sqcap \forall \textit{measureOf}.\text{SpatialTemporal}$$
$$\text{MeasureValue} \sqcap \exists \textit{belongsTo}.\text{SpatialTemporal} \sqsubseteq \exists \textit{measureOf}.\top \sqcap \textit{measureOf} \doteq \textit{belongsTo}$$

```
(213) (defconcept Angle-Measure
        :is (:and Geometry-Measure
                  (:the unit Angle-Unit))
        :implies (:all measure-of Angle))
```

DL:
$$\text{AngleMeasure} \equiv \text{GeometryMeasure} \sqcap = 1\textit{unit} \sqcap \forall \textit{unit}.\text{AngleUnit}$$
$$\text{AngleMeasure} \Rightarrow \forall \textit{measureOf}.\text{Angle}$$

Considering the definitions above, we could have an `Angle-Measure` instance asserted to be linked to a `Spatial-Temporal` instance through a `belongs-to` relation. Then this relation needs to be recognized as being of type `measure-of`, which is then used as a precondition in a conceptual implication to determine that the instance it is linked to is an `Angle`.

9.3.7. Constraint propagation on individuals

Since concept and relation definitions can be expressed by double concept and relation implications, we could include the previous two cases in a larger inference service, which could be called Constraint Propagation. What the service needs to do is whenever an individual is modified by either asserting a property or a relation on it, it needs to apply all TBox implications to this individual or other individuals related to it.

Moreover, the implications need to be performed through a forward-chaining inference mechanism, since there is no explicit target that can be inquired. And they need to be

applied repeatedly, so that all applicable implications are performed automatically on all existing instances.

9.4. Portability to other Logic Systems

The previous sections described the specific features of the Loom logic system used in implementing our approach to NLU and showed their equivalents in generic Description Logic terms. The question arises whether these features are unique to Loom, or whether they can be found in other existing systems, facilitating the implementation of our approach in those systems.

The question can be split into two separate parts: the first considers whether the logic language features used in our system are present in other systems; the second part investigates whether the inference services necessary for implementing our approach can be found in alternative logic systems.

The question will be examined in this section in the context of two of the state of the art logic systems currently available: PowerLoom and Racer.

9.4.1. PowerLoom

PowerLoom is the successor of the Loom system, developed at the Information Sciences Institute of University of Southern California. PowerLoom departs from Loom mainly by providing a more powerful language (Chalupsky & al, 2003). PowerLoom uses a variant of the Knowledge Interchange Format (KIF) as input language, to express definitional and assertional content. KIF provides constructs for the expression of arbitrary sentences in First-Order predicate calculus (Genesereth & Fikes, 1992). It also includes a number of built-in theories, like sets, lists, and numbers. Besides KIF, PowerLoom continues to provide a number of Description Logic-specific constructs, like special forms for defining concepts, relations, and functions, and stating number, domain, and range restrictions. Some of these are discussed below.

9.4.1.1. Logic language

PowerLoom includes all conceptual language features of Loom. Since the input language is First-Order Logic, it models concepts through unary predicates. Thus concept conjunction and disjunction, as well as full negation can be expressed as the corresponding Algebra expressions on unary predicates.

```
(214) (defconcept Infinite-Object
        :is-primitive (:and Open-Object
                            (:not Finite-Object)))
```

DL: InfiniteObject ⊑ OpenObject ⊓ ¬FiniteObject

```
PL: (defconcept Infinite-Object (?x)
      :=> (and (Open-Object ?x)
               (not (Finite-Object ?x))))
```

PowerLoom also provides facilities to specify concept equality and concept inclusion, either through definitions or through separate logic assertions. PowerLoom does not have

a specific construct for trigger rules, but multiple concept definitions are allowed, so that the need for them has disappeared.

Relations in PowerLoom are also modeled through predicates. The predicates are not restricted to binary form, so arbitrary arity relations can be modeled directly. PowerLoom provides constructs to define relation equalities and relation inclusions, as well as specific constructs for functional relations. Multiple definitions are also allowed so again there is no need for specific trigger rules. PowerLoom does not distinguish between concepts and relations at the language level, so relation disjunction and relation negation can also be expressed similarly to concepts.

Concept and relation hierarchies can be expressed in PowerLoom either directly in the definitions, or through logic implication and equivalence assertions. PowerLoom does not provide specific language constructs for expressing role composition, but the input language does allow fully quantified variables. In the PowerLoom examples below free variables are universally quantified by default.

```
(215) (implies (:compose belongs-to belongs-to)
               belongs-to)
```

DL: $belongsTo \circ belongsTo \Rightarrow belongsTo$

```
PL: (assert (=> (and (belongs-to ?x ?y) (belongs-to ?y ?z))
                (belongs-to ?x ?z))))
```

Similarly, universal and existential role restrictions can be expressed using universal and existential variables, as seen below:

```
(216) (defconcept Pair-Of-Numbers
        :is (:and Pair
                  (:all element-role Natural-Number)))
```

DL: $\text{PairOfNumbers} \equiv \text{Pair} \sqcap \forall elementRole.\text{NaturalNumber}$

```
PL: (defconcept Pair-Of-Numbers ((?x Pair))
      :<=> (forall ((?y Natural-Number))
                   (element-role ?x ?y)))
```

The example above also illustrates how concept restrictions can be expressed directly through typed variable declarations. The definition is equivalent to:

```
(217) PL: (defconcept Pair-Of-Numbers (?x)
            :<=> (and (Pair ?x)
                      (forall (?y) (and (element-role ?x ?y)
                                        (Natural-Number ?y))))))
```

Some of the Description Logic features are supported in PowerLoom through built-in second-order logic predicates. Most such predicates are given PowerLoom definitions, based on a primitive `holds` predicate. The `holds` predicate asserted on a set of arguments has the meaning that the first argument is a predicate name that holds for the rest of the arguments. For example:

```
(218) PL: (assert (holds belongs-to ?x ?y))
```

is equivalent to:

```
(219) PL: (assert (belongs-to ?x ?y))
```

The second-order nature of the `holds` relation comes from the fact that all its arguments can be variables, thus allowing quantification over predicate names.

Thus transitive, inverse, and symmetric roles have built-in second-order predicates with the definitions:

```
(220) PL: (defrelation transitive ((?r RELATION))
            :=> (=> (and (holds ?r ?x ?y) (holds ?r ?y ?z))
                    (holds ?r ?x ?z)))

      PL: (deffunction inverse ((?r BINARY-RELATION) ?i)
            :=> (<=> (holds ?i ?y ?x) (holds ?r ?x ?y))

      PL: (defrelation symmetric ((?r RELATION))
            :=> (<= (holds ?r ?x ?y) (holds ?r ?y ?x))))
```

Similarly, domain and range restrictions have corresponding second-order predicates with definitions:

```
(221) PL: (defrelation domain ((?r RELATION) (?d CONCEPT))
            :=> (=> (holds ?r ?i ?v) (holds ?d ?i)))

      PL: (defrelation range ((?r RELATION) (?rng CONCEPT))
            :=> (=> (holds ?r ?i ?v) (holds ?rng ?v)))
```

An example of use is shown below:

```
(222) (defrelation congruent-to
        :is (:and equal-to
                  (:domain Finite-Object)
                  (:range Finite-Object)))
```

congruentTo $\sqsubseteq$ *equalTo*

DL: $\exists$*congruentTo*.$\top \sqsubseteq$ FiniteObject $\sqcap \forall$*congruentTo*.FiniteObject

FiniteObject $\sqcap \exists$*equalTo*.FiniteObject $\sqsubseteq \exists$*congruentTo*.$\top \sqcap$ *congruentTo* $\doteq$ *equalTo*

```
      PL: (defrelation congruent-to (?x ?y)
            :<=> (and (equal-to ?x ?y)
                      (domain congruent-to Finite-Object))
                      (range congruent-to Finite-Object)))
```

As seen in a similar example above, domain and range restrictions can be also specified more directly through typed variable declarations:

```
(223) PL: (defrelation congruent-to ((?x Finite-Object)
                                     (?y Finite-Object))
            :=> (equal-to ?x ?y))
```

PowerLoom also supports unqualified number restrictions through built-in predicates. Three predicates are defined for the three cases: `range-cardinality-lower-bound` for `:at-least`, `range-cardinality-upper-bound` for `:at-most`, and `range-cardinality` for `:exactly`. An example of use is given below:

```
(224) (defconcept Measure-Value
        :is-primitive (:and Abstraction Nondecomposable-Object
                            (:at-most 1 value)
                            (:at-most 1 unit)))
```

DL: MeasureValue $\sqsubseteq$ Abstraction $\sqcap$ NonDecomposableObject $\sqcap \leqslant 1$*value* $\sqcap \leqslant 1$*unit*

```
PL: (defconcept Measure-Value (?x)
      :=> (and (Abstraction ?x)
               (Nondecomposable-Object ?=x)
               (range-cardinality-upper-bound value ?x 1)
               (range-cardinality-upper-bound unit ?x 1)))
```

`:relates` constructs can be expressed in PowerLoom directly by asserting the new relation on the variables corresponding to the fillers of existing relations:

```
(225) (implies (:and Result (:some cause Being&Having)
                            (:some effect Spatial-Temporal))
               (:relates results-in cause effect))
```

DL: Result $\sqcap \exists$*cause*.Being & Having $\sqcap \exists$*effect*.SpatialTemporal
$\Rightarrow \forall$*cause*.$\exists$*resultsIn*.$\top \sqcap$ *cause* $\circ$ *resultsIn* $\doteq$ *effect*

```
PL: (assert (=> (and (Result ?x)
                     (exists ((?y Being&Having))
                             (cause ?x ?y))
                     (exists ((?z Spatial-temporal))
                             (effect ?x ?z)))
                (results-in ?y ?z)))
```

Finally role-value maps can be represented in PowerLoom using the equality built-in predicates:

```
(226) (implies (:and Get (:some actee Element)
                         (:some property Thing))
               (:same-as actee property))
```

DL: Get $\sqcap \exists$*actee*.Element $\sqcap \exists$*property*.Thing $\Rightarrow$ *actee* $\doteq$ *property*

```
PL: (assert (=> (and (Get ?x)
                     (exists ((?y Element)) (actee ?x ?y))
                     (exists ((?z Thing))
                             (property ?x ?z)))
                (= ?y ?z)))
```

As mentioned before, PowerLoom also includes the concrete domain of numbers as a built-in theory, so our use of numbers can be transferred without modifications. Additionally, PowerLoom provides support for non-monotonic logic, in the form of retraction of assertions. Retractions were not used in our implementation because they were not working in conjunction with instance unification in Loom. However they have been found to be desirable in a few cases, the most important of which is reference resolution.

9.4.1.2. Semantic representation and contexts

PowerLoom supports ABox reasoning, by allowing the user to declare individuals (called objects) and assert predicates on them. Similarly to Loom, normal objects in PowerLoom work under the Unique Name Assumption. This fact raises a problem for role-value maps and for unification of objects (see next section), since asserting an equality predicate on two such objects is always false by definition. However, similarly to Loom, PowerLoom also provides skolem objects. Our semantic representation for natural language needs to be modeled using this specific kind of objects, so that equality results in the desired unification effect.

PowerLoom also has a mechanism of separate assertional workspaces called contexts. Unlike Loom, in PowerLoom contexts can also partition the definitional language. Similar to Loom, contexts can be built in a hierarchy, where each context can inherit from several others. Thus the use of contexts in our system can be transferred without changes to PowerLoom.

9.4.1.3. Reasoning services

As stated in Chalupsky & Russ (2002), "PowerLoom uses a form of natural deduction to perform inference and it combines a forward and backward chaining reasoner to do its work." While PowerLoom "is not a complete theorem prover for first-order logic, it has various reasoning services … that go beyond the capabilities of a traditional first-order theorem prover." Thus, PowerLoom supports concept subsumption and has a classifier that uses technology derived from Loom (Chalupsky & al, 2003). Moreover the classifier also works on arbitrary arity predicates, thus also providing role specialization (called relation classification). PowerLoom also provides a service for individual realization, called instance classification. Both classification services can be called explicitly by the user.

PowerLoom also has capabilities to provide the other services used in our approach. It automatically checks concepts and relations for satisfiability, and objects for consistency, signaling when there are problems.

One important question arises however, which is which of these reasoning services can be used in forward-chaining inference. PowerLoom provides explicit calls for some of these services, like concept classification and instance realization. However in a simple test, usual implications asserted in the definition of concepts and relations do not seem to be applied in forward-chaining mode (at least in the current 3.0.2 version).

PowerLoom does provide a special construct for defining forward-only (=>>) implications (as well as backward-only (<<=)), and it also has a special application call for forward-chaining rules. Rules declared with this construct are indeed applied when forward-chaining rules are called explicitly, and the manual states that they can be triggered also by the backward-chaining reasoning mechanism. A test seemed to show that this mechanism is powerful enough for our needs. Although, since PowerLoom is not a complete reasoner, it is hard to know whether it will always give the desired result.

9.4.2. Racer

Racer is one of the state of the art Description Logic systems currently available, developed at the Technical University of Hamburg. According to Haarslev & Möller (2004) "The RACER system is a knowledge representation system that implements a highly optimized tableau calculus for a very expressive Description Logic. It offers reasoning services for multiple TBoxes and for multiple ABoxes as well."

9.4.2.1. Logic language

Racer includes many of the features used in our implementation, but not all. It includes concept conjunctions, disjunctions and full negations (Haarslev & Möller, 2004). It supports several types of concept axioms. It allows the expression of both concept inclusions and concept equalities, and provides specific constructs for full and primitive definition of concepts. It also provides axioms for explicit expression of concept disjointness, so there is no need to implement it using negation:

```
(227) (defconcept Non-Conscious-Thing
        :is-primitive (:and Simple-Thing
                            (:not Conscious-Being)))
```

DL: $\mathsf{NonConsciousThing} \sqsubseteq \mathsf{SimpleThing} \sqcap \neg \mathsf{ConsciousBeing}$

```
Racer: (define-primitive-concept Non-Conscious-Thing
         Simple-Thing)
       (disjoint Non-Conscious-Thing Conscious-Being)
```

Since there is no limit on the number of concept inclusions or equalities that can be stated for any given concept, trigger rules are again not needed.

Racer also provides constructs similar to Loom for expressing universal and full existential role quantifications, as well as unqualified and qualified number restrictions (even if the latter were not used in our approach).

Racer's role language is however more restricted than Loom's. Racer supports role transitivity and functional roles (Haarslev & Möller, 2004) through special declarations:

```
(228) (implies (:compose belongs-to belongs-to)
               belongs-to)
```

DL: $\mathit{belongsTo} \circ \mathit{belongsTo} \Rightarrow \mathit{belongsTo}$

```
Racer: (transitive belongs-to)
```

```
(229) (defrelation count
        :is-primitive relation
        :characteristics :single-valued)
```

DL: $\mathit{count} \sqsubseteq \mathit{relation}$
$\exists \mathit{count}.\top \sqsubseteq \leqslant 1 \mathit{count}$

```
Racer: (define-primitive-role count :parents (relation))
       (functional count)
```

When the functionality restriction applies only in the context of a specific concept, role number restrictions can be used:

```
(230) (defconcept Collective-Thing
        :is (:and Thing
                  (:the count Collective)))
```

DL: $\mathsf{CollectiveThing} \equiv \mathsf{Thing} \sqcap (= 1\mathit{count}) \sqcap \exists \mathit{count}.\mathsf{Collective}$

```
Racer: (define-concept Collective-Thing
         (and Thing (exactly 1 count)
                    (some count Collective)))
```

Inverse and symmetric roles are also supported in Racer through special declarations. The inverse declaration provides the name for the inverse role. The symmetric declaration just states that the role has itself as inverse:

```
(231) (defrelation part-of
        :is-primitive belongs-to)
      (defrelation has-part
        :is (:inverse part-of))
```

DL: $\mathit{partOf} \sqsubseteq \mathit{belongsTo}$
$\mathit{hasPart} \equiv \mathit{partOf}^{-}$

```
Racer: (define-primitive-role part-of :parents (belongs-to))
       (inverse part-of has-part)
```

```
(232) (defrelation opposite-to
        :is-primitive relation
        :characteristics (:single-valued :symmetric))
```

DL: $\mathit{oppositeTo} \sqsubseteq \mathit{relation}$
$\mathit{oppositeTo} \equiv \mathit{oppositeTo}^{-}$

```
Racer: (define-primitive-role opposite-to
         :parents (relation))
       (symmetric opposite-to)
```

Racer does not support role conjunctions, role disjunctions, or role negations. None of these is a serious problem for our approach, since the last two were not used, and the first one was used only to add domain and range restrictions. Racer has specific constructs for specifying domain and range restrictions, thus making role conjunctions unnecessary:

```
(233) (defrelation count
        :is-primitive (:and relation
                            (:domain Thing)
                            (:range Count-Value)))
```

DL: $\mathit{count} \sqsubseteq \mathit{relation}$
$\exists \mathit{count}.\top \sqsubseteq \mathsf{Thing} \sqcap \forall \mathit{count}.\mathsf{CountValue}$

```
Racer: (define-primitive-role count :parents (relation))
       (domain count Thing)
       (range count Count-Value)
```

Racer provides role hierarchies, but they are restricted to atomic roles. Role hierarchies can be specified either in the form of primitive role declarations with specification of role parents (see previous examples), or by explicit atomic role implication clauses (see

example below). The fact that one can specify multiple parents for a given role could also be used to express simple cases of role conjunction, if needed:

```
(234) (defrelation r1
        :is-primitive (:and r2 r3))
```

DL: $r1 \sqsubseteq r2 \sqcap r3$

```
Racer: (define-primitive-role r1)
       (implies-role r1 r2)
       (implies-role r1 r3)
```

Racer however does not allow the expression of full role definitions, nor general (non-atomic) role inclusions. The primitive role declarations are also unique per role, multiple declarations being signaled as errors (Haarslev & Möller, 2004).

The absence of role definitions or general role inclusions constitutes a serious problem for the implementation of our approach, because some of the inferences needed in our approach rely on the logic system being able to classify a role in a specific configuration as a more specific one. For instance we can take example (222) repeated below:

```
(235) (defrelation congruent-to
        :is (:and equal-to
                  (:domain Finite-Object)
                  (:range Finite-Object)))
```

In this case, our approach relies on the system being able to recognize that an `equal-to` relation asserted between two instances of type `Finite-Object` is also a `congruent-to` relation. This inference is important for recognizing the semantic equivalence of sentences like:

```
(236) a) The angles are equal.

      b) The angles are congruent.
```

One solution to the problem is to convert this definition to a pair of role inclusions:

```
(237) (defrelation congruent-to
        :is-primitive (:and equal-to
                            (:domain Finite-Object)
                            (:range Finite-Object)))
      (implies (:and equal-to
                     (:domain Finite-Object)
                     (:range Finite-Object))
               congruent-to)
```

However, since Racer does not allow general role inclusions either, the second construct needs to be changed into an inclusion on concepts:

```
(238) (implies (:and Finite-Object
                     (:some equal-to Finite-Object))
               (:and (:some congruent-to Finite-Object)
                     (:same-as equal-to congruent-to)))
```

However this can lead to the necessity to use some other constructs, like the role-value map above, which might not be expressible in Racer.

Another possible way to replace role definitions is to 'reify' relations by changing them into relationship concepts (Borgida & Brachman, 2003). Then the links between the new concept and previous role domain and range fillers have to be modeled through new roles, with restricted semantics, which do not need relation definitions. As an example, if we take the `congruent-to` relation in example (235) again, we can change it into:

```
(239) (defconcept Congruent-To
        :is (:and Equal-To
                  (:the role1 Finite-Object)
                  (:the role2 Finite-Object)))
```

DL: CongruentTo
$\equiv$ EqualTo $\sqcap$ =*1role1* $\sqcap$ $\exists$*role1*.FiniteObject $\sqcap$ =*1role2* $\sqcap$ $\exists$*role2*.FiniteObject

```
Racer: (define-concept Congruent-To
         (and Equal-To
              (exactly 1 role1)
              (some role1 Finite-Object)
              (exactly 1 role2)
              (some role2 Finite-Object)))
```

As seen in this example, the new concept can be readily modeled in Racer. This transformation would however lead to extensive changes over the entire knowledge base, as well as on the process of compositional building of semantic representations. For instance, in the example above `Equal-To` also needs to be changed into a concept, and so do all relations above it and all concepts that use it in their definitions.

Two constructs used in our approach that Racer lacks are role composition and role-value maps. In cases where role composition is used to model role transitivity, we can dispense with it by asserting the transitivity property directly on the role (see example (228) above). However in most cases role composition was used in the system to assert agreement on chains of functional roles. Such cases cannot be modeled in Racer. Moreover it has been proved that agreements on role chains used in conjunction with general inclusion axioms cause subsumption to become undecidable (Nebel, 1991), even when restricted to functional chains. However agreements on role chains play an important role in our approach, allowing us to assert equivalence among various ways natural language has to express the same meaning. It is not clear at this point whether there is an alternative way to model the same phenomenon. The Loom-specific `:relates` construct was translated into Description Logic using the same agreements on role chains, so it falls into the same category.

Racer also provides a rich language for the concrete domain of numbers, so our limited use of integer values can be translated with no difficulty. Roles with concrete domain values are called attributes in Racer and need to be declared using included together with the appropriate value type in the signature declaration:

```
(240) (defrelation cardinality
        :domain Set
        :range Number
        :characteristics :single-valued)
      (defconcept Pair
        :is (:and Set (:the cardinality 2)))
```

cardinality ⊑ *relation*

DL: Set ⊓ ∃*cardinality*.T ⊑⩽ 1*cardinality* ⊓ ∀*cardinality*.Number

Pair ≡ Set ⊓= 1*cardinality* ⊓ ∃*cardinality*.2

```
Racer: (signature :atomic-concepts (Pair Set)
                  :attributes ((integer cardinality)))
```

9.4.2.2. Semantic representation and contexts

Similarly to PowerLoom, Racer supports ABox reasoning, together with a rich set of reasoning services on them. Thus, modeling semantic representations through individuals is not a problem. However instances in Racer work under the Unique Name Assumption, and no "skolem" instances are provided, so merging or equating instances is not allowed. One way to avoid merging of instances could be to make copies of them. Thus when we need to merge instance `i1` to instance `i2`, one could create a new instance `i3`, assert on it all predicates that hold on `i1`, and then add on it all assertions that hold on `i2`. Racer does provide a `describe-individual` service that collects the logic result of all assertions that had been made on an instance, so its result could be used in creating the new instance. In remains to be seem whether this operation is too expensive to be used extensively during the parsing process.

A more serious problem is that of separate contexts. Racer does provide reasoning with multiple ABoxes, however they cannot inherit from one another. Racer does allow the user to make copies, or 'clones' of existing ABoxes, but this is not enough, since in our approach child contexts usually inherit from two parent contexts. One way to implement this behavior in Racer could be to create a clone of one of the two parent ABoxes, and then assert in it all assertions that hold in the second parent. Racer does provide services to collect all assertions from a given ABox, so that could be used in the process. However it is probable that the process would be prohibitively expensive for extensive use in each step of the parsing process.

9.4.2.3. Reasoning services

As quoted at the beginning of section 9.4.2, Racer used an optimized tableau calculus as the base for its reasoning services. As a result, unlike Loom and PowerLoom, Racer provides a complete reasoner over the accepted language (Haarslev & Möller, 2003). This would constitute a considerable advantage over the other systems, since it guarantees that all true facts are actually derivable by the system.

Racer provides most of the inference services used in our approach. Thus it provides concept classification, concept satisfiability (as TBox coherence check), instance realization, and instance satisfiability (as ABox coherence check). These services also work in a forward-chaining mode at the ABox level. The only service we use that it does

not provide is that of role specialization, since role definitions are not allowed in the logic language. Ways to avoid using them were discussed in section 9.4.2.1.

Chapter 10 Conclusions and Future Work

The dissertation presented an innovative approach to natural language understanding, combining a Feature Structure unification formalism for syntax and a Description Logic framework for semantics. A detailed description of the system components was provided, as well as an example of how they work together to build compositionally the semantic representation of natural language sentences.

The thesis also examined a number of problems occurring in students' explanations expressed in natural language, and showed how these problems can be solved elegantly within the framework of the proposed approach.

The systems' classification performance was evaluated on real classroom data and was compared to a Naïve Bayes approach. A portability study to a new semantic domain was also performed and an evaluation of the time effort and the volume of changes was provided.

Finally, the thesis provided a high-level examination of the Description Logic features and services needed to support the proposed approach to natural language understanding, and discussed the possibility of port it to other state of the art logic systems.

10.1. Semantic repair mechanism

An analysis of the corpus of student explanations from our last evaluation shows that there are a number of classes of problems that the current system is has difficulties with. Some cases were mentioned in section 7.2. A few of them are analyzed in more detail below.

10.1.1. Implicit references

One class of problems that currently pose a challenge to the system is that of contextual or implicit references. Here is an example from our recent study:

(241) `It's an isosceles triangle, so the base angles are congruent.`

The problem with this sentence is that there is no explicit reference stating what angles it is talking about. However the use of the definite article '`the`' requires that the angles in question must be previously known, or related somehow to elements previously mentioned. Below are a few other examples that show the same problem:

(242) a) The shape is an isosceles triangle so the two bottom angles are equal.

b) The two angles that make up a right angle are complementary because the sum is equal to 90 degrees.

c) Three adjacent angles make up the larger angles so you must add them to find the measure.

10.1.2. Malformed sentences

Another difficult case is that when the semantic constraints of the elements linked by the grammar rules do not match entirely. One such case is that of metonymy, presented in section 3.2.4. The current solution, presented in section 5.3.9, lacks generality, as it only deals with a number of specific cases. Here is a case not currently covered:

(243) Ray HF bisects measure of angle GHR producing two congruent angles.

The problem here is that a ray bisects an angle, not its measure. So the measure is used instead of the angle. More examples of the malformed sentences, either on the syntactic or on the semantic level, are given below:

(244) a) Isosceles triangles have isosceles angles.

b) In a triangle their is 180 degrees.

c) When you as two adjacent angles they add up the measure our the big angle.

d) The measure of an angle formed by adjacent angles is equal to the sum of the measured of those adjacent angles.

e) Add the two part adjacent parts of the angle to get the sum of the entire angle.

10.1.3. Incompleteness in knowledge base coverage

A third category of problems in our recent study was generated by the incomplete coverage of our lexicon and/or our knowledge base. Thus there were a number of explanations that were perfectly valid from a syntactic and semantic point of view, but the system was not able to classify them. A few examples showing the missing element are:

(245) a) The sum of two adjacent angles is the third angle.

b) Adjacent angles along a line add up to 180 degrees.

c) All angles combined in a triangle are equal to 180 degrees.

d) The adjacent angles sum to the cause angle.

10.1.4. Semantic vicinity conceptual search

A possible solution that has the potential to cover the problems above, and possibly others, is to try to repair the semantic representation after the parsing ends. Thus, in cases where a full semantic representation could not be built, but the process ends with several fragments, we could start a search process that looks in the semantic vicinity of the concepts involved, looking for valid ways to connect the fragments.

Thus, in case (241) above the search would find from the definition of concept `Triangle` that it has angles as vertices. In example (243) it could show that a possible argument to a `Bisection` concept is an `Angle`. In the cases shown by (245) the search could find ways to connect the semantic fragments on both sides of the unknown element through the most plausible semantic link.

A preliminary test showed however an important difficulty. The semantic representation fragments that need to be connected consist both of a number of different instances, together with all properties and relations asserted on them. In absence of more information, a semantic vicinity search is probable to find numerous different ways to connect the two given semantic representations. In order to limit the possibilities, the search has to know which instances to try to connect in each representation. This information could be available as a result of the parsing process. However, it is not easy to detect it and pass it to the semantic repair process.

Bibliography

Aleven, V.; Koedinger, K. R.; and Cross, K. (1999). Tutoring answer-explanation fosters learning with understanding. In Lajoie, S. P. and Vivet, M. eds. *Artificial Intelligence in Education, Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration, Proceedings of AIED-99*, 199-206. Amsterdam, Netherlands: IOS Press.

Aleven, V; Popescu, O; and Koedinger, K. R. (2001). Towards tutorial dialog to support self-explanation: Adding Natural Language Understanding to a Cognitive Tutor. In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.), *Artificial Intelligence in Education: AI-ED in the Wired and Wireless Future*, *Proceedings of AI-ED 2001,* 246-255. Amsterdam, Netherlands: IOS Press.

Aleven, V; Popescu, O; and Koedinger, K. R. (2002). Pilot-testing a tutorial dialogue system that supports self-explanation. In Cerri A, Gouarderes G, Paraguacu F eds. Proceedings of Sixth International Conference on Intelligent Tutoring Systems, ITS2002. Berlin, Germany: Springer Verlag; 2002. 246-255.

Alshawi, H. (ed) (1992). *The Core Language Engine*, MIT Press, Cambridge, Massachusetts.

Altmann, G.; Steedman, M. (1988). Interaction with Context during Human Sentence Processing, *Cognition*, 30:191-238.

Anderson, J. R.; Corbett, A. T.; Koedinger, K. R.; and Pelletier, R. (1995). Cognitive tutors: Lessons learned. In *The Journal of the Learning Sciences*, 4:167-207.

Anderson, J. R.; and Lebiere, C. (1998). *The Atomic Components of Thought*. Hillsdale, NJ: Erlbaum.

Baader, F.; Calvanese, D.; McGuinness, D.L.; Nardi, D.; and Patel-Schneider, P.F. (eds.) (2003). The Description Logic Handbook, Theory, Implementation, and Applications, Cambridge University Press.

Baader, F.; and Nutt, W. (2003). Basic Description Logics. In Baader & al (eds.). *The Description Logic Handbook, Theory, Implementation, and Applications,* Cambridge University Press.

Bateman, J.; Magnini, B.; and Rinaldi, F. (1994). The Generalized Italian, German, English Upper Model. In *Proceedings of the ECAI-94 Workshop on Implemented Ontologies*, Amsterdam, pp. 35--45.

Borgida, A.; Brachman, R.J. (2003). Conceptual Modeling with Description Logics. In Baader & al (eds.). *The Description Logic Handbook, Theory, Implementation, and Applications,* Cambridge University Press.

Bloom, B. S. (1984). The 2 sigma problem: the search for methods of group instruction as effective as one on one tutoring. *Educational Researcher,* 13, 4-16.

Bresnan, J. (2001). *Lexical-Functional Syntax.* In Textbooks in Linguistics series. Blackwell.

Brill, D (1993). *Loom™ Reference Manual* for Loom version 2.0, Information Sciences Institute, University of Southern California.

Brown, J.S.; Burton, R.; and de Kleer, J. (1982). Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III. In Sleeman, D. and Brown, J.S. (Eds.), *Intelligent Tutoring Systems*, Academic Press.

Chalupsky, H.; MacGregor, R.; Russ, T.A. (2003). *PowerLoom™ Manual*. Powerful knowledge representation and reasoning with delivery in Common-Lisp, Java, and C++, Information Sciences Institute, University of Southern California.

Chalupsky, H.; and Russ, T.A. (2002). WhyNot: Debugging Failed Queries in Large Knowledge Bases. In *Proceedings of the Fourteenth Innovative Applications of Artificial Intelligence Conference (IAAI-02),* Menlo Park, AAAI Press.

Chi, M.T.H.; Siler, S.; Jeong, H.; Yamauchi, T.; & Hausmann, R.G. (2001). Learning from tutoring. In *Cognitive Science*, 25:471-533.

Cohen, J. (1960). A coefficient of agreement for nominal scales. In *Educational and Psychological Measurement,* 20, 37-46.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. In *Psychological Bulletin,* 70, 213-220.

Dowding, J.; Gawron, J. M.; Appelt, D.; Bear, J; Cherny, L.; Moore, R.; and Moran, D. (1993). Gemini: A Natural Language System for Spoken-Language Understanding. In *Proceedings of ARPA Workshop on Human Language Technology,* Princeton, New Jersey, Morgan Kaufmann Publisher, 43-48.

Fass, D. (1988). Metonymy and Metaphor: What's the Difference? In Proceedings of COLING 1988, Budapest, 177-181.

Foltz, P. W.; Laham, D.; and Landauer, T. K. (1999). Automated Essay Scoring: Applications to Educational Technology. In *Proceedings of EdMedia '99.*

Genesereth, M. R.; and Fikes, R. E. (1992). Knowledge Interchange Format Version 3 Reference Manual, Logic-92-1, Stanford University Logic Group.

Glass, M. (1997). Some phenomena handled by the Circsim tutor version 3 input understander. *In Proceedings of FLAIRS97*, Daytona Beach, Florida.

Glass, M. (2000). Processing Language Input in the CIRCSIM-Tutor Intelligent Tutoring System. In *Proceedings of AAAI 2000 Fall Symposium on Building Dialogue Systems for Tutorial Applications, AAAI Press, California, 74-80.*

Glass, M. (2001). Processing Language Input in the CIRCSIM-Tutor Intelligent Tutoring System. In Moore, J.D. & al (eds). *Artificial Intelligence in Education. Proceedings of AIED 2001, IOS Press, 210-221.*

Haarslev, V.; and Möller, R. (2003). Racer: A Core Inference Engine for the Semantic Web. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003)*, located at the 2nd International Semantic Web Conference ISWC 2003, Florida.

Haarslev, V.; and Möller, R. (2004). *RACER User's Guide and Reference Manual Version 1.7.19*, Technical University of Hamburg.

Heffernan, N. T.; Koedinger, K. R. (2000). Intelligent Tutoring Systems are Missing the Tutor: Building a More Strategic Dialog-Based Tutor. In *Proceedings of AAAI 2000 Fall Symposium on Building Dialogue Systems for Tutorial Applications.*

Jordan, P. W.; Rosé, C. P.; and VanLehn, K (2001) Tools for Authoring Tutorial Dialogue Knowledge. In *Proceedings of Artificial Intelligence in Education.*

Kay, M. (1986). Algorithm schemata and data structures in syntactic processing. In Grosz, B.J. & al. (eds.). *Readings in Natural Language Processing*. Morgan Kaufmann, Los Altos, California, 35-70.

Kimball, J. (1973). Seven Principles of Surface Structure Parsing in Natural Language. In *Cognition*, 2-1, 15-47.

Koedinger, K. R.; Anderson, J. R.; Hadley, W. H.; and Mark, M. A. (1997). Intelligent tutoring goes to school in the big city. In *International Journal of Artificial Intelligence in Education*, 8:30-43.

Landauer, T. K.; Foltz, P. W; and Laham, D. (1998). Introduction to Latent Semantic Analysis. In *Discourse Processes*, 25, 259-284.

Langley, C. (2003). *Domain Action Classification and Argument Parsing for Interlingua-Based Spoken Language Translation*, PhD Thesis, Carnegie Mellon University.

Lavie, A. (1995). *A grammar based robust parser for spontaneous speech*. Ph.D. Thesis, Carnegie Mellon University.

Lavie, A.; Langley, C.; Waibel, A.; Pianesi, F.; Lazzari, G.; Coletti, P.; Taddei, L.; and Balducci, F. (2001). Architecture and Design Considerations in NESPOLE: a Speech Translation System for E-commerce Applications. In *Proceedings of the 1st International Human Language Technology Conference (HLT-2001),* San Diego, California.

MacGregor R. (1991). Using a description classifier to enhance deductive inference. In *Proceedings of the Seventh IEEE Conference on AI Applications*, 141-147. Miami, Florida.

Mitchell, T. (1997). *Machine Learning*, McGraw Hill.

Möller, R.; and Haarslev, V. (2003). Description Logic Systems. In Baader & al (eds.). *The Description Logic Handbook, Theory, Implementation, and Applications,* Cambridge University Press.

Nebel, B. (1991). Terminological Cycles: Semantics and computational properties. In Sowa, J.F. (ed.). *Principles of Semantic Networks*, Morgan Kaufmann, Los Altos.

Nyberg, E.; Kamprath, C.; and Mitamura, T. (1998). The KANT Translation System: From R&D to Large-Scale Deployment, in *Localization Industry Standards Association Newsletter*, 2:1.

Nyberg, E.; and Mitamura, T. (2000). The KANTOO Machine Translation Environment. In *Proceedings of AMTA* 2000.

Pollard, C. J.; and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press, Chicago, Illinois, 197.

Rosé, C. P.; and Lavie, A. (1999). *LCFlex: An efficient robust left-corner parser*, User's manual, University of Pittsburgh.

Rosé, C. P. (2000): Syntactic framework for semantic interpretation. In *Proceedings of the Workshop on Linguistic Theory and Grammar Implementation 2000*, Birmingham, Great Britain.

Rosenkrantz, D.J. and Lewis, II, P.M. (1970). Deterministic left corner parsing. In *Proceedings of 11th Symposium on Switching and Automata Theory*, IEEE Computer Society, 139-152.

Shieber, S.M.; Uszkoreit, H.; Robinson, J.; and Tyson, M. (1983). *The formalism and Implementation of PATR-II*. SRI International, Menlo Park, California.

Tomita, M. (1988). *The generalized LR parser/compiler*. User's guide, Carnegie Mellon University.

Wiemer-Hastings, P.; Wiemer-Hastings, K.; and Graesser, A. C. (1999). Improving an intelligent tutor's comprehension of students with latent semantic analysis. In Lajoie, S.P. & Vivet, M. (eds.) *Artificial Intelligence in Education, Open Learning Environments: New Computational Technologies to Support Learning, Exploration, and Collaboration, Proceedings of AIED-99*, 535-542. Amsterdam, Netherlands: IOS Press.

Witten, I. H.; and Frank, E. (2000). *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco.