
Processing Paraphrases and Phrasal Implicatives in the Bridge Question-Answering System

UNDERGRADUATE HONORS THESIS

SYMBOLIC SYSTEMS PROGRAM
STANFORD UNIVERSITY

Karl Pichotta

June 8, 2008

Processing Paraphrases and Phrasal Implicatives in the Bridge Question-Answering System

Karl Pichotta

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, for conferral of Honors.

Lauri Karttunen
Consulting Professor
Department of Linguistics
Stanford University
(Primary Advisor)

Date

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, for conferral of Honors.

Tracy Holloway King
Consulting Associate Professor
Symbolic Systems Program
Stanford University
(Second Reader)

Date

Acknowledgements

I am grateful to many people for their assistance with this project. The research in this thesis was conducted alongside Matt Paden, whose contributions were invaluable. Lauri Karttunen and Tracy King have been extremely helpful in guiding this thesis on the right track; Valeria de Paiva and the entire Natural Language Theory and Technology group at PARC were exceedingly helpful and supportive. In addition, I must thank Todd Davies for his writing advice.

Contents

1	Introduction	1
2	The Bridge Question-Answering System and Background	2
2.1	Introduction to the Bridge System	2
2.2	Processing from Character String to Abstract Knowledge Representation	3
2.3	Concepts, Contexts, and the AKR	5
2.3.1	Example AKRs	7
2.3.2	More In-Depth Example AKRs	10
2.4	Mechanics of Inference	14
2.4.1	The Question-Answering Task	14
2.4.2	ECD Mechanics	15
2.5	Types of Inferences	18
2.5.1	Factives	18
2.5.2	Implicatives	19
3	Paraphrases and Phrasal Implicatives	21
3.1	Paraphrases	21
3.2	Phrasal Implicatives	24
3.2.1	Conditional Lexical Properties	25
3.2.2	The Characterization of Verbs and Nouns	27
3.2.3	Implication or Implicature?	27
4	Implementation and Operation	28
4.1	The Rewrite System	28
4.1.1	Introduction to the Rewrite System	28
4.1.2	Simple Example Rewrite Rules	31
4.2	Rewrite Rule for Phrasal Implicatives	32
4.3	Examples of Operation	35
5	Conclusions and Possibilities for Further Research	37

1 Introduction

The task of automated natural-language question answering and, more broadly, natural language understanding, is a difficult one. The performance of textual inference (determining exactly what can and cannot be justifiably inferred from a text) by current specialized systems is certainly not nearly as good as that of natural speakers of the language in question. There are many individual phenomena in natural language to account for, and as systems' abilities to handle these phenomena increases, so too does the performance of systems in the tasks of textual inference and question-answering improve. The **Bridge** system, created at Palo Alto Research Center (PARC), is one such question-answering system: it accepts, as input, sentences in a natural language¹ and performs various textual inferences upon them for the ultimate end of being able to answer questions about the input text. For a broad overview of the system, see Bobrow et al. (2007).

This thesis will outline a number of constructions in English that bear certain presuppositions and entailments in English. For example, consider the following pairs of sentences:

- (1) a. The cat managed to scratch the girl.
b. The cat scratched the girl.

- (2) a. The cat had the nerve to scratch the girl.
b. The cat scratched the girl.

It is clear that (1b) can be safely inferred from (1a) and, similarly, (2b) can be inferred from (2a): we, as speakers of the English language, find these inferences obvious, and almost trivial. However, such phenomena must be explicitly categorized and encoded into a natural language understanding system if a computer is to systematically and consistently be able to recognize such inferences when encountered.² In addition to outlining a few such systematic phenomena in English, this thesis will explain, in some detail, the implementation of the processing of these phenomena in the **Bridge** system.

Section 2 introduces the **Bridge** question-answering system, its means of representation of language-based knowledge, and the basic mechanics behind its inferential reasoning system. Section 2.5 gives an introduction to factives and implicatives, linguistic constructions relevant to the remainder of the thesis. Section 3 gives a discussion of paraphrases and their processing. It also contains an explanation of phrasal implicatives, a class of linguistic constructions which systematically induce certain inferences. The processing of a relatively common class of phrasal implicatives in the **Bridge** system is the focus of section 4, which gives the code used and traces through its execution. Finally, section 5 summarizes the thesis and gives possibilities for future work.

¹The particular grammar the system has is for the English language, but there is no reason why the system could not instead use a grammar and lexicon for Japanese, Chinese, or German and accept sentences in that language as input. In fact, large-scale grammars for these languages have already been developed.

²The inferences in both (1) and (2) are driven by the same phenomenon, implicativity, which is explained in detail in §2.5.

2 The Bridge Question-Answering System and Background

2.1 Introduction to the Bridge System

PARC's Bridge system is a large-scale question-answering system. It takes a passage of text, consisting of one or more sentences, and a query sentence. The system then, given a query sentence, returns YES, NO, or UNKNOWN, depending on whether the query sentence follows from the passage text, its negation follows from the text, or its truth value cannot be determined from the passage text, respectively. Consider the following three pairs of sentences:

- (3) a. Pat ate a steak.
b. Pat ate meat.

- (4) a. Todd did not buy anything.
b. Todd bought a van.

- (5) a. Cassandra was shoved.
b. Lyle shoved Cassandra.

Example (3) shows a valid inference: given sentence (3a), we can conclude (3b); thus, the system should answer YES given (3a) as a passage sentence and (3b) as a query. Similarly, since, given (4a), we can conclude the negation of (4b), the system should answer NO given the sentences in (4) as a passage and query, respectively. The final example shows a lack of entailment: given (5a) as a passage, we do not know whether (5b) is true or false (we do not know who shoved Cassandra); the system should, therefore, answer UNKNOWN given (5) as a passage and query. The task at hand is one of Entailment and Contradiction Detection (ECD): the system should return YES if a query is entailed by the passage, NO if a query contradicts the passage, and UNKNOWN if a query is neither entailed by the passage nor contradicts it. Figure 1 shows the interface to the Bridge system.

The lexicon that is used by the system is derived from three sources (Crouch and King, 2005). It incorporates the freely available WordNet, which provides a coherent, wide-ranging ontology through lists of hypernyms and synonyms (Fellbaum, 1998). It is through WordNet that the lexicon comes to note that, for example, *red* can be a color, or *cat* can be a type of animal.³ The second source incorporated is VerbNet (also freely available), which enables the lexicon to map syntactic roles (such as object or subject) to semantic roles (such as Agent or Theme) (Kipper et al., 2000). For example, the subject of *run* in the sentence *Jim ran to the store* (in this case, *Jim*) plays the semantic role of Theme; the lexicon gets such information from VerbNet. Exactly what semantic roles are and the details of how VerbNet treats them are

³I write *can be* instead of *is* here because most words have numerous senses, and both *red* and *cat* have a number of disparate meanings (consider “a primary color” vs. “a communist” for *red*, or “a feline” vs. “a devotee of jazz” for *cat*). The handling of such lexical ambiguity is discussed in more detail below.

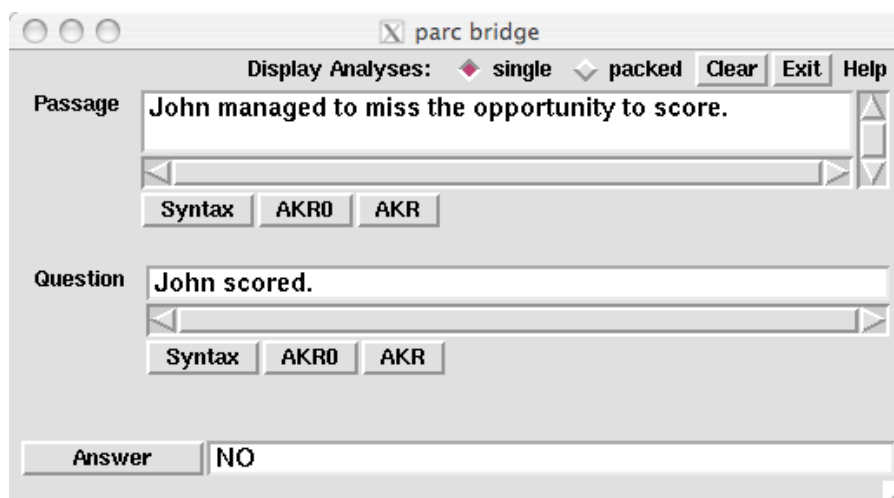


Figure 1: Bridge interface window

beyond the scope of this thesis. It suffices to note that the word *dog* has identical semantic roles in the sentences *The dog chased the cat* and *The cat was chased by the dog*, though it has a different syntactic function in each. It is important, for the purposes of answering meaningful questions about text, that the word *dog* play the same role in both of these sentences; otherwise, the system would be too sensitive to syntactic changes that have negligible effect on the meaning of a sentence.⁴ VerbNet and its system of roles are based on the Levin verb classes; for details, see Levin (1993).

The third source that the lexicon draws from is the XLE syntactic lexicon, which contains the grammatical subcategorization frames for English verbs, adjectives, and nouns. A subcategorization frame specifies what types of arguments a predicate might take. For example, among the 23 subcategorization frames present for the verb *see* are V-SUBJ and V-SUBJ-OBJ. The first corresponds to a sentence like *I can see* (where *see* is intransitive), and the second describes a transitive usage such as that in *I can see the light*.

2.2 Processing from Character String to Abstract Knowledge Representation

In order to answer questions about a text, we must operate on a representation of the text that is abstract enough so as to not be affected by changes in wording. So, for example, if we know *The president did not talk for long*, and we ask the question *Was the president's speech brief?*, the answer should be YES, despite the fact that the two sentences are worded differently. Below is a brief sketch of the stages a sentence will go through before inferences are drawn on it.

The input to the **Bridge** system is a string of characters which presumably represents a well-formed English sentence. The string is parsed using the XLE parser (Crouch et al., 1993-2007).

⁴For example, the syntactic roles of a sentence and its passivized equivalent may differ, but the semantic roles are identical. In both of the given examples, the dog is the Agent of the verb *chase* and cat is the Theme.

The XLE parser uses the hand-crafted rules of a Lexical-Functional Grammar (LFG) in order to output a functional-structure (f-structure) for an input sentence. An f-structure represents the grammatical roles and relationships of constituents of the sentence. For example, an f-structure for the sentence *I didn't comb my hair* contains the fact that the verb *comb* has the subcategorization frame V-SUBJ-OBJ (meaning the verb takes a subject and an object). It further contains these two constituents (*I* and *my hair*, respectively). It also contains more detailed information about the sentence (for example, that *I* is a first person singular pronoun, that the mood of the sentence is indicative, that the main verb is used actively, not passively, etc.). For a detailed explanation of LFG and f-structures, see Kaplan and Bresnan (1982) and Dalrymple (2001). An example f-structure – for the sentence *Mary hopped* – is given in Figure 2. The details of f-structures are beyond the scope of this thesis; it suffices to note that Figure 2 contains quite a bit of information about the subject (e.g. that *Mary* is a proper name for female human), about the verb (e.g. that *hop* in this sentence has the subcategorical frame V-SUBJ, that is, it behaves intransitively), and about the sentence itself (e.g. it is in the past tense, is not passive, and is indicative).

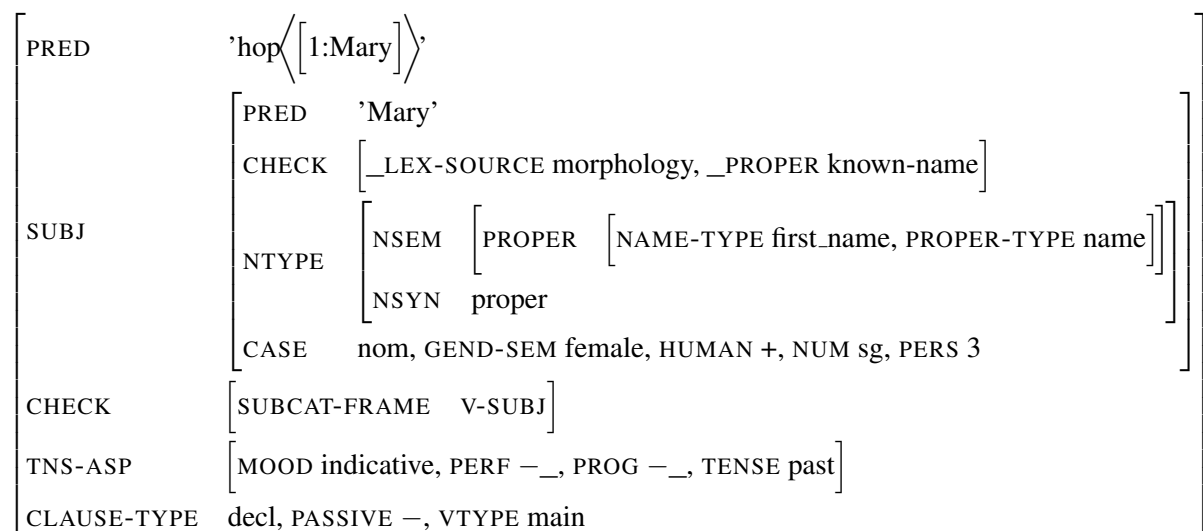


Figure 2: f-structure for the sentence *Mary hopped*.

This syntactic analysis, as performed by the XLE parser, is notable because, instead of disambiguating a sentence immediately and picking a probable “most likely” candidate reading, it will retain many possible readings of a sentence for processing later on down the line. It is for this reason that the output of the XLE parser is often referred to as a PACKED representation – because numerous possible readings of a sentence are “packed” into a single, compact representation (Maxwell and Kaplan, 1991). This is a useful thing to do for efficient natural language processing. Suppose the relevant passage sentence is the canonical example of structural ambiguity given in (6).

- (6) Jim saw the girl with the telescope.

Now, consider the following two queries (which we treat as possible conclusions of the above passage sentence):

- (7) a. Jim used the telescope to see the girl.
- b. Jim saw a girl who had a telescope.

Ideally, we would like to say that both of the above query sentences follow from the passage (both (7a) and (7b) follow logically from a different, but justified, reading of (6)). If we commit to one of the two immediately obvious readings of the passage sentence too early in the process, we will be able to get only one of the conclusions from (7).

The output of this syntactic analysis is then analyzed semantically (Crouch and King (2006), Crouch (2005), Dalrymple (2001)). In this stage of processing, purely syntactic facts that do not affect the meaning of the sentence are dropped. Further, it is at this point that information from WordNet and VerbNet is incorporated into the representation. That is, the ontological relationships between words to be found in the lexicon are added to the system's representation of the sentences at this step, as are the roles of participants.

The final stage of processing a sentence or a set of sentences is to convert the semantic output into an ABSTRACT KNOWLEDGE REPRESENTATION (AKR). The AKR level is sufficiently abstract so as to allow textual inferences of the sort required to detect entailment and contradiction. The composition of the AKR is outlined in more depth in §2.3.

2.3 Concepts, Contexts, and the AKR

The AKR is based in large part on what is called Textual Inference Logic 2 (or TIL2), which contains both CONCEPTS and CONTEXTS. For a more in-depth coverage of this system, see de Paiva et al. (2007) and Bobrow et al. (2005). In short, the system uses *concepts* to represent entities, providing intensional descriptions of entities and events described in the sentence. It also uses *contexts*, used to make statements about the existence of entities satisfying the descriptions (McCarthy, 1993). Both concepts and contexts are described below.

The AKR does not use variables to represent entities and events described by a sentence. It instead uses concept constants. Suppose we wish to produce an AKR for the sentence *The man ran*. We do not create an arbitrary variable x and a predication $\text{man}(x)$; we instead create a particular subconcept called, say, `man:1` (though this concept could be named anything: `man:2`, `c2`, or `sally` would all be just as valid names). We also create a subconcept to represent the running event, say, `run:2`. We then state facts about these particular subconcepts.⁵ For more information on this process and the reasoning behind it, see Condoravdi et al. (2001).

There are several relations between concepts. The first is the relation `subconcept(x, y)`, which states that x is a subconcept of y . So, for example, the fact

```
subconcept(armadillo, animal)
```

⁵Note that this process is very similar to Skolemization in relational logic, in which an existentially quantified variable x not in the scope of any universal quantifiers is replaced with an unused constant c (so that, for example, $\exists xPx$ gets Skolemized to Pc). The sentence *the man ran* can be thought of as stating *there exists a man and there exists a running event such that the man participated in it*. When viewed this way, it makes sense to create Skolem constants representing the entities *man* and *run* and declare that the entities denoted by these two Skolem constants are related in a certain way (namely, that the man participated in the running event).

obtains, since any instance of the concept `armadillo` is an instance of the concept `animal`. This is a useful relation to reason with because the lexicon used by the system already contains a fairly extensive ontological hierarchy based on the hypernym relation between words from WordNet.

The second type of relation between concepts is the `role` relation. The relation `role(r, x, y)` states that concept `y` plays role `r` to concept `x`. So, for example, in the sentence *The dog ate the biscuit*,

```
role(Agent, eat:2, dog:1)
```

holds, since `dog:1` is the Agent of the concept `eat:2` (i.e. the dog was engaging in that particular eating event).

The AKR also uses *contexts*, which combine to form a “frame” in which it makes sense to talk about propositions related to concepts being true or false. A context can be viewed as a set of propositions or facts. The need for contexts becomes clear when examining a sentence like *Calvin believes Goldbach’s conjecture is true*. In such a sentence, two contexts will be created, a top-level context (which we will conventionally refer to as τ) and another context representing what Calvin believes. The top-level context τ represents the speaker’s commitment (if something is true in this context, then the speaker believes it to be true). We assume anything true in this context is true; conversely, anything false in this context is false. The proposition “Goldbach’s conjecture is true” holds in the context representing Calvin’s belief, but is not necessarily true in the top-level context τ . A context, again, is to be thought of as a set of propositions; we can represent exactly which propositions hold in a certain context by their inclusion in this set. Contexts thus keep track of exactly where a certain fact obtains and where it does not. For example, something can be true in the context representing Calvin’s belief, but not necessarily true outside that context; note, however, that, in the outer context τ , it is still the case that Calvin believes the proposition represented within his belief context, namely, *Goldbach’s conjecture is true*.

Truth and existence are represented in this system by *instantiability*: a declaration that a particular instance of a concept is *instantiable* in a certain context states the truth of an assertion within that context. A declaration that a particular instance of a concept is *uninstantiable* within a certain context states the falsity of an assertion within that context. For example, with the sentence *The man ran*, both the concept Skolem created for man and for the running event will be instantiable in the top-level context τ , because we conclude both the man existed and the running event existed. However, in the sentence *The man didn’t run*, the Skolem constant created for man will be instantiable in τ , but the one for run will be uninstantiable (since no such running event – one with the man as its Agent – took place).

A third type of relation (in addition to the `subconcept` and `instantiable` relations) places event Skolems in time. For example, in the past-tense sentence *Susie fought furiously*, the following obtains:

```
temporalRel(startsAfterEndingOf, Now, fight:1)
```

where `fight:1` is the Skolem constant representing the fighting event that Susie engaged in. The above states that the time referred to by `Now` (which is to be interpreted as the time of the

speaker’s utterance of the sentence) starts after the ending of the event referred to by the Skolem constant `fight:1`. This is, in this particular example, a way of encoding the fact that the fighting event happened in the past; such facts about the relative placement of events in time are useful for temporal reasoning.

2.3.1 Example AKRS

We now examine some examples. Figure 3 is the AKR produced by the system for the sentence *The cat sat*.

```

Conceptual Structure:
  subconcept (sit:2, [sit-1, sit-2, sit_down-1, sit-4, model-3, ...])
  role (Theme, sit:2, cat:1)
  subconcept (cat:1, [cat-1, guy-1, cat-3, cat-o'-nine-tails-1, ...])
  role (cardinality_restriction, cat:1, sg)
Contextual Structure:
  context (t)
  top_context (t)
  instantiable (cat:1, t)
  instantiable (sit:2, t)
Temporal Structure:
  temporalRel (startsAfterEndingOf, Now, sit:2)

```

Figure 3: AKR for *The cat sat*

The first `subconcept` line states that `sit:2` (the Skolem constant corresponding to the occurrence of *sit* in the sentence) is a subconcept of one of the elements in the list enclosed by brackets. Each element in this list corresponds to one of the different senses of the word *sit* that could be used here – the given `subconcept` fact states that the `sit:2` Skolem constant is a subconcept of one of these senses. That is, the list enclosed in brackets after the Skolem constant is a list of possible hypernyms of the word *sit*, each hypernym being represented as a list of synonyms. These different senses are derived from WordNet. The second `subconcept` fact states a similar assertion, but for the `cat:1` Skolem constant instead (the senses represented in the list of possible ancestor concepts corresponding to the senses of “feline”, “young man”, “spiteful female gossip”, “a whip with nine cords”, and so on). Each of these senses is represented internally as a pointer to a set of synonyms corresponding to this particular sense of the word.

Referring to a subconcept such as `sit:2` as a Skolem constant as described in footnote 5 is slightly misleading: `sit:2` is more than an arbitrary symbol introduced to eliminate some sort of existential quantifier. A subconcept constant like `sit:2` is comprised of a list of possible hypernyms, each of which is a group of synonyms (a *synonym set*, or *synset*) which are best thought of abstractly as describing a “cloud” of possible meanings around a certain sense of the word *sit*. This representation of the meaning of each word as a disjunction of hypernym “clouds” is to be contrasted with approaches such as Montague grammar (Montague, 1970), where the meaning of a lexical item is represented by an arbitrary constant, such as *sit'*. A subconcept is, in

this formulation, a fundamentally intensional construction which is very much dependent upon the meaning of the word or concept it ultimately represents.

That all of these possible senses of each of these words are preserved throughout the process is consistent with the philosophy of “packing” a given sentence’s representation and not disambiguating too early in the process. This corresponds to the way that we, as speakers of the language, behave. If, for example, Bob states *I need a match*, we will draw a different conclusion about what Bob needs if we see him holding an unlit cigar than we will if we see him holding a single sock. Similarly, we do not want to disambiguate word sense too early in the process, just as we do not wish to disambiguate syntactically too early in the process. To do so may be to commit to an incorrect reading of a sentence; the correct reading can be determined only by downstream processing.

The two `role` facts are straightforward: since, semantically, in the sentence, *cat* is the Theme of *sit*, the fact `role(Theme, sit:2, cat:1)` is present (this information is stored in the lexicon). The second `role` fact states that there is only one cat; that is, the cardinality of the `cat:1` concept is restricted to be singular. Facts with `role cardinality_restriction`, such as this, provide information about the cardinality of a particular subconcept; for example, *cat* has a cardinality of `sg`, since it is singular; *cats* will have a cardinality of `pl`, on account of its being plural. Keeping track of the cardinalities of subconcepts is not particularly important for the types of inferences discussed in this thesis, but it is necessary for certain types of quantifier-based reasoning. For example, with the following two sentences:

- (8) a. At least three people know my Social Security Number.
- b. At least two people know my Social Security Number.

(8a) should imply (8b), but the converse implication should not hold. In order to perform such quantifier-based reasoning, it is important to keep track of the cardinality of subconcepts (in this case, we are keeping track of the size of the set of people who know my Social Security Number, which will be a fact about the subconcept representing *people*).

The contextual structure for the AKR in Figure 3 consists of nothing more than the top-level context, called `t`, in which both `cat:1` and `sit:2` are instantiable. This means that in the top-level context (the context representing the speaker’s truth commitments), there exists a cat (represented by concept `cat:1`), and there exists a sitting event (represented by concept `sit:2`). That this particular cat is the one doing the sitting is represented elsewhere, in the `role(Theme, sit:2, cat:1)` fact.

Finally, the fact that the sitting event happened in the past (since the verb *sit* occurs in the past tense in the sentence) is represented by the

```
temporalRel(startsAfterEndingOf, Now, sit:2)
```

fact, which states that `Now`, corresponding to the time of the speaker’s utterance of the sentence, starts after the ending of the sitting event. That is, the event represented by the Skolem constant `sit:2` ended before the present.

Consider again the sentence *Jim saw the girl with the telescope* from (6). Figure 4 shows the AKR for this sentence. For the most part, this example is similar in structure to the last. There

```

Choice Space:
  xor(A1, A2) iff 1
Conceptual Structure:
  subconcept (see:5, [see-1, understand-2, witness-2, visualize-1, ...])
  A1: role (prep (with), see:5, telescope:25)
      role (Stimulus, see:5, girl:13)
      role (Experiencer, see:5, Jim:1)
      subconcept (Jim:1, [male-2])
      alias (Jim:1, [Jim])
      role (cardinality_restriction, Jim:1, sg)
      subconcept (girl:13, [girl-1, female_child-1, daughter-1, girlfriend-2, ...])
  A2: role (prep (with), girl:13, telescope:25)
      role (cardinality_restriction, girl:13, sg)
      subconcept (telescope:25, [telescope-1])
      role (cardinality_restriction, telescope:25, sg)
Contextual Structure:
  context (t)
  top_context (t)
  instantiable (Jim:1, t)
  instantiable (girl:13, t)
  instantiable (see:5, t)
  instantiable (telescope:25, t)
Temporal Structure:
  temporalRel (startsAfterEndingOf, Now, see:5)

```

Figure 4: AKR for *Jim saw the girl with the telescope.*

are a number of `subconcept` facts, giving the list of possible hypernyms for each entity in the sentence. There are `role` facts, some giving the semantic roles of the verb *see* and some giving cardinality restrictions of entities. This example, unlike the last, has an `alias` fact, which states that its first argument is a Skolem constant for an entity with a proper name.⁶

This example differs from the last, however, in that the AKR in Figure 4 contains a CHOICE SPACE to manage the sentence’s structural ambiguity. There are two `role` facts labeled A1 and A2. Notice the line

```
xor(A1, A2) iff 1
```

which states that either the fact marked with A1 or the fact marked with A2 holds (but not both). In the above, 1 is synonymous with what is typically denoted \top in propositional logic (a constant that is always true), so the statement is equivalent to the simpler statement `xor(A1, A2)`. Intuitively, A1 and A2 describe the difference between the conceptual structures given by either of the readings for the sentence. That is, the fact marked A1:

```
A1: role(prepare(with), see:5, telescope:25)
```

describes the reading where the PP *with the telescope* attaches high and Jim used the telescope to see the girl. The fact marked A2:

```
A2: role(prepare(with), girl:13, telescope:25)
```

corresponds to the alternate reading, where the PP attaches low and the girl has the telescope. This is a very compact representation of the sentence’s natural structural ambiguity – the vast majority of the AKRs representing either reading of the sentence is shared; the two AKRs differ only by a single fact (meaning, roughly, that *with* applies to either *girl* or *see*). The many facts common to both AKRs are stored only once, and a relatively minimal choice space representation is used to manage structural ambiguity. The syntax, semantics, AKR, and ECD (Cf §2.4.2) all use the same packing mechanism to maintain efficiency of representations and processing time. Throughout processing, the representations never need to be unpacked.

2.3.2 More In-Depth Example AKRs

We now examine the AKRs for two similar, but crucially different, sentences. Figure 5 contains the AKR for the sentence *Bob knows that Scruffy ate*. The conceptual structure for this sentence is similar to previous examples. There are a number of `subconcept` facts, each stating possible synsets (synonym sets) for words, each of which defines a distinct sense of the word. There are `role` facts, some giving the semantic roles for the verbs, some giving cardinality restrictions. There are various `alias` facts, each stating that its first argument has a proper name.

⁶This is useful for examples where the same entity can be referred to by different phrases. For example, if we know *Barack Obama won the Virginia primary*, we should be able to properly determine that the sentence *Obama won at least one primary* follows from the first, even though the same man is referred to differently in the two sentences. In this case, *Barack Obama* and *Obama* are both aliases of the same Skolem constant, representing the subconcept of the person named Barack Obama who won the Virginia primary. In addition, alias facts are used to differentiate between the subconcepts representing different proper names. For example, *Edward* and *Robert* will both have the subconcept synset representing a human male, and are differentiated only by their respective alias facts.

```

Conceptual Structure:
  subconcept (know:2, [know-1, know-2, acknowledge-6, roll_in_the_hay-1, ...])
  role (Agent, know:2, Bob:0)
  role (Theme, know:2, ctx (eat:6))
  subconcept (Bob:0, [male-2])
  alias (Bob:0, [Bob])
  role (cardinality_restriction, Bob:0, sg)
  subconcept (eat:6, [eat-1, eat-2, feed-6, consume-5, eat-5, corrode-1])
  role (Agent, eat:6, Scruffy:5)
  subconcept (Scruffy:5, [entity-1])
  alias (Scruffy:5, [Scruffy])
  role (cardinality_restriction, Scruffy:5, sg)
Contextual Structure:
  context (t)
  context (ctx (eat:6))
  top_context (t)
  context_lifting_relation (veridical, t, ctx (eat:6))
  context_relation (t, ctx (eat:6), crel (Theme, know:2))
  instantiable (Bob:0, t)
  instantiable (Scruffy:5, t)
  instantiable (eat:6, t)
  instantiable (know:2, t)
  instantiable (Scruffy:5, ctx (eat:6))
  instantiable (eat:6, ctx (eat:6))
Temporal Structure:
  temporalRel (temporallyContains, know:2, Now)
  temporalRel (startsAfterEndingOf, Now, eat:6)

```

Figure 5: AKR for *Bob knows that Scruffy ate*

The contextual structure for this sentence is more complicated than that for *The cat sat*. There are two contexts – first, the top-level context τ , representing speaker commitment; second, there is the context representing what Bob knows, referred to in the AKR as $\text{ctx}(\text{eat}:6)$. These two contexts are related to each other. Note first the presence of the line

```
context_relation( $\tau$ ,  $\text{ctx}(\text{eat}:6)$ ,  $\text{crel}(\text{Theme}, \text{know}:2)$ )
```

which states that the two contexts are related to each other by the verb *know*, and that the relation comes from the fact that $\text{ctx}(\text{eat}:6)$ is the Theme of $\text{know}:2$. The second fact stating a relationship between the two concepts is

```
context_lifting_relation( $\text{veridical}$ ,  $\tau$ ,  $\text{ctx}(\text{eat}:6)$ )
```

which states that anything that is instantiable in the second context $\text{ctx}(\text{eat}:6)$ is also instantiable in the first context τ . This behavior comes from the semantics of the word *know* in English: by assumption, if X knows φ then φ is true, regardless of who X is.

This last relation between the two contexts manifests itself in the set of instantiable facts present. Note first the two facts

```
instantiable( $\text{Scruffy}:5$ ,  $\text{ctx}(\text{eat}:6)$ )
instantiable( $\text{eat}:6$ ,  $\text{ctx}(\text{eat}:6)$ )
```

state that two Skolem constants are instantiable in the $\text{ctx}(\text{eat}:6)$ context. Intuitively, this states that the entity described by $\text{Scruffy}:5$ exists in the context representing what Bob knows, and that the eating event described by $\text{eat}:6$ (which has $\text{Scruffy}:5$ as an Agent) also exists in this context. Note now these two facts:

```
instantiable( $\text{Bob}:0$ ,  $\tau$ )
instantiable( $\text{know}:2$ ,  $\tau$ )
```

which state that the entity described by $\text{Bob}:0$ and the event described by $\text{know}:2$ exist in the outer context τ , representing speaker commitment. This makes sense: the speaker is committed to the existence of a person named Bob who knows something.

The final two instantiability facts:

```
instantiable( $\text{Scruffy}:5$ ,  $\tau$ )
instantiable( $\text{eat}:6$ ,  $\tau$ )
```

are the instantiability facts from the $\text{ctx}(\text{eat}:6)$ context “lifted” to the τ context. This occurs because of the presence of the `context_lifting_relation` fact, stating that anything true in the inner context is also true in the outer context. Because of the last two instantiability facts, we can properly determine that a sentence like *Scruffy ate* follows logically from *Bob knows that Scruffy ate*, because the AKR for the latter sentence contains within it these two facts stating that the concepts representing Scruffy and the eating event are both instantiable in the outermost context τ .

Consider now Figure 6, which contains the AKR for the sentence *Bob thinks that Scruffy ate*. This AKR is almost identical to the one in Figure 5, on account of the sentences differing only

Conceptual Structure:

```
subconcept (think:2, [think-1, think-2, think-3, remember-1, intend-1, ...])
role (Theme, think:2, ctx (eat:7))
role (Experiencer, think:2, Bob:0)
subconcept (Bob:0, [male-2])
alias (Bob:0, [Bob])
role (cardinality_restriction, Bob:0, sg)
subconcept (eat:7, [eat-1, eat-2, feed-6, consume-5, eat-5, corrode-1])
role (Agent, eat:7, Scruffy:6)
subconcept (Scruffy:6, [entity-1])
alias (Scruffy:6, [Scruffy])
role (cardinality_restriction, Scruffy:6, sg)
```

Contextual Structure:

```
context (t)
context (ctx (eat:7))
top_context (t)
context_relation (t, ctx (eat:7), crel (Theme, think:2))
instantiable (Bob:0, t)
instantiable (Scruffy:6, t)
instantiable (think:2, t)
instantiable (Scruffy:6, ctx (eat:7))
instantiable (eat:7, ctx (eat:7))
```

Temporal Structure:

```
temporalRel (startsAfterEndingOf, think:2, eat:7)
temporalRel (temporallyContains, think:2, Now)
temporalRel (startsAfterEndingOf, Now, eat:7)
```

Figure 6: AKR for *Bob thinks that Scruffy ate*

in the main verb. Note, however, that while X knows φ does entail φ , it is not the case that X thinks ψ entails ψ . So, where the AKR in Figure 5 had a `context_lifting_relation` fact, there is no such fact in this AKR. Thus, in contrast to the AKR in Figure 5, there is no fact stating that the eating event represented by `eat : 7` is instantiable in the outer context t . Given the query *Did Scruffy eat?*, the system will answer UNKNOWN because there is no fact stating the instantiability of `eat : 7` as there was in the AKR in Figure 5.

2.4 Mechanics of Inference

2.4.1 The Question-Answering Task

Bridge is a Question-Answering system. The system takes as input two snippets of text: a passage and a query. Typical Question-Answering systems return two values: something like TRUE if the query text is entailed logically by the passage text, and FALSE if the query text is not entailed by the passage. For example, the PASCAL Recognizing Textual Entailment Challenge (Dagan et al., 2006) is a much-noted benchmark of this task. According to this bivalent TRUE/FALSE question-answering task, if we have the passage/query pair

(9) P. Jon has a pet cat and a pet dog.

Q. Jon has a pet dog.

the system should return TRUE, as the passage (9P) entails the query (9Q). If, on the other hand, we have the passage/query pair

(10) P. Jon has a pet cat and a pet dog.

Q. Jon has no pets.

we should get FALSE, since the passage (10P) does not entail the query (10Q). Similarly, under the PASCAL system, we get FALSE given the passage/query pair

(11) P. Jon has a pet cat and a pet dog.

Q. Jon has a pet dalmatian.

since, again, the passage does not entail the query.

The Bridge system, however, operates differently. Instead of a two-valued TRUE or FALSE answer, Bridge gives a three-valued YES/NO/UNKNOWN answer. YES is returned if all possible situations verifying the passage also verify the query (that is, it is not possible for the passage to be true and the query to be false). NO is returned if it is not possible for the query to be true if the passage is true; that is, if we assume the passage to be true, then the query cannot be true. UNKNOWN is returned if it is unclear whether the query is true given the passage; that is, if we assume the passage to be true, the query could be either true or false. So, for the passage/query pairs given in (9), (10), and (11), the Bridge system will answer YES, NO, and UNKNOWN, respectively. That is, the system distinguishes between a passage entailing the contradiction of the query, as in (10), and a passage neither entailing the query nor contradicting it, as in (11), in which Jon might have a dalmatian, but he also might have some other type of dog.

The system is not limited to simple YES/NO/UNKNOWN answers. Just as in a typical resolution-based logic programming language one can put a variable in a query and get the value of this variable verifying the query, one can insert some number of anaphors into the query sentence which will get “filled in” if the system answers YES. For example, consider the passage/query pair

- (12) a. Liz was attacked by a bear.
b. Who did a bear attack?

The Bridge system will answer YES to this passage/query pair (meaning that the question can be answered positively). The interrogative pronoun *who* is treated like the word *someone*, so a positive answer to the question means that someone is found who was attacked by a bear. The answer will further specify that the person who makes the query true is Liz (i.e. will state *someone = Liz*).

2.4.2 ECD Mechanics

The task at hand, then, is, given a passage and a query, to determine whether the query is entailed by the passage, whether the query contradicts the passage, or whether neither of these is the case (as noted above, this is referred to as Entailment and Contradiction Detection, or ECD). Such work is performed on the AKR level. The general mechanism is as follows.

- (i) Create an AKR for the passage and an AKR for the query.
- (ii) At this point, the system will go through and consider alignments between Skolem constants from the query and Skolem constants from the passage. This means that, for example, if the passage contains a fact

`role(Theme, punch:4, man:3)`

and the query contains a fact

`role(Theme, punch:1, man:5)`

(both corresponding to something like *the man was punched*) then, if the subconcepts `punch:1` and `punch:4` each have subconcept facts stating they are appropriately similar, and `man:3` and `man:5` also have appropriate subconcept facts, then we align the two `role` facts given above. They must have similar subconcept facts because we do not want to align two Skolem constants that cannot possibly mean the same thing. Recall that subconcept facts contain the possible senses of meanings that a subconcept has.

- (iii) Determine specificity relations among aligned Skolem constants – for example, if *cat* aligns to *animal* in the Passage/Query pair *The cat died/ An animal died*, then it is determined that *animal* is more general than *cat*, as the lexicon has *animal* marked as a hypernym of *cat*.

- (iv) Given an alignment between concepts (and their relative specificity), remove all appropriate facts from the query AKR.
- (v) If all facts can be removed from the query AKR, then the query follows from the passage, and we can answer YES.
- (vi) If there is some `instantiateable` Skolem constant that aligns with some `uninstantiateable` Skolem constant, then there is a contradiction, and we can answer NO, as the query contradicts the passage somehow.
- (vii) If, for every valid alignment, there are facts remaining in the query AKR, then we answer UNKNOWN, since we can say neither that the query follows from the passage nor that it contradicts the passage.

This procedure is sound with respect to our intuitions about entailment and contradiction. A query q is entailed by a passage p if and only if everything that q says is also said by p . For example, if our query q is the sentence *It rained yesterday* and the passage p is the sentence *It rained yesterday, but it did not freeze*, then everything that q contains semantically is also contained in p . This corresponds to step (v) above. Conversely, if q is the sentence *It rained yesterday, but it did not freeze*, and p is the sentence *It rained yesterday*, then there is a lack of entailment from p to q , since q says more than p does. This corresponds to step (vii) above, representing a lack of entailment, but not contradiction. A query q contradicts a passage p if there is something in q whose negation is contained in (or is entailed by) p . For example, if q is *It did not rain yesterday* and p is *It rained yesterday*, then some fact will be uninstantiable in the query AKR that is instantiable in the passage AKR. That is, something will be false in the query AKR that is true in the passage AKR. This corresponds to step (vi) above. Note that the uninstantiable concept could occur in the passage AKR, with the instantiable concept in the query AKR; what is important is the presence of a contradiction, and not where the logical negation occurs.

Step (ii) creates an alignment between the concepts in the passage and in the query. After this step, it is important to perform monotonicity-based specificity calculations (step (iii)). Consider the passage/query pair

(13) P. Every boy saw a black cat.

Q. Every small boy noticed a cat.

This inference is, indeed, valid, and the system should answer YES when given (13) as input. In step (ii) of the process, *boy* in (13P) will align with *small boy* in (13Q), *saw* in (13P) will align with *noticed* in (13Q), and *black cat* in (13P) will align with *cat* in (13Q). The semantics of the quantifier phrase *every NP_i VP_j* in English are such that NP_i is in a downward monotone environment and VP_j is in an upward monotone environment. That is, given a sentence of the form *every NP_i VP_j*, we can replace the NP_i with a more specific NP and VP_j with a less specific VP and get a true sentence. Thus, (13Q) does indeed follow logically from (13P).

In order for the **Bridge** system to calculate inferences like this, it must perform specificity calculations on aligned concepts. That is, we calculate that the NP *small boy* is more specific

than the NP *boy*. Similarly, we note that *noticed* is less specific than *saw* (that is, *notice* is a hypernym of at least one sense of *see*) and that *cat* is less specific than *black cat*, so the VP *noticed a cat* is less specific than *saw a black cat*. After performing these specificity calculations between aligned concepts, then, we can remove aligned facts (step (iv) above) and get the desired conclusion.

This specificity calculation step is fairly nuanced. With the passage/query pair

(14) P. Not every small boy noticed a cat.

Q. Not every boy saw a black cat.

Bridge will answer YES, but with the pair

(15) P. Not every boy noticed a black cat.

Q. Not every boy noticed a cat.

Bridge will answer UNKNOWN (every boy may have seen some a white cat). This is so because the monotonicity behavior of the quantifier *every* switches when it is negated. Similar to this, if given input

(16) P. Ted drove from Dallas to Houston.

Q. Ted drove.

the system should answer YES since (16P) entails Ted performing a less specific action. Similarly, given

(17) P. Ted did not drive.

Q. Ted drove from Dallas to Houston.

the system should answer NO, since (16P) entails Ted *not* performing a more specific action than is given. Thus, specificity calculation on aligned concepts is crucial to the task of Question-Answering.

The above procedure allows us to draw certain types of inferences by treating the passage and query AKRs differently. Since, in order for a passage p to imply a query q , it is *not* the case that we need to eliminate all facts from p 's AKR, we can expand p 's AKR to contain certain types of inferences we would like to draw without necessarily expanding q in the same way. For example, from the passage *Anderson was killed*, we would like to be able to conclude that *Anderson died*. We can therefore instruct the system to, in the case of lethal verbs (*kill*, *murder*, etc.), when they occur in the passage, also add the fact that the lethal verb's Theme (in the above example, Anderson) died. This allows us to conclude that *Anderson died* follows from *Smith killed Anderson*. Note, however, that we should not make this expansion when a lethal verb occurs in the query, since it will never be the case that something like *Anderson was killed* will follow from the passage *Anderson died*.

2.5 Types of Inferences

There are certain types of inferences to be drawn that are critical to a question-answering system like *Bridge*. As speakers of English, these inferences may seem, at times, trivial, but processing them systematically is a necessity for any sort of natural language understanding system. Consider, for example, the following sentences:

- (18) a. The sky darkened.
b. The sky got darker.

From (18a) we would like to conclude (18b) holds (and vice versa). Note that calculating the veridicity of (18b) from (18a) is not a matter of calculating truth in the world, but rather about calculating truth with respect to the commitments of the author of the passage being processed. That is to say, we assume (18a) was stated by an author and that the author is committed to the veridicality of the statement. We can then conclude that (18b) holds *according to the commitments of the speaker*, but not necessarily in the real world. Below are introduced two types of constructions – factives and implicatives – that induce certain inferences.

2.5.1 Factives

Consider the following inferences:

- (19) a. Smith acknowledged that foul play occurred. \rightsquigarrow Foul play occurred.
b. Sally forgot that computers crash sometimes. \rightsquigarrow Computers crash sometimes.
c. Samuel knew that the brakes had given out. \rightsquigarrow The brakes had given out.

The inferences illustrated in (19) are based on PRESUPPOSITIONS. These inferences are valid because, for example, if Smith *acknowledges* X is the case, then we know that, according to the author of the sentence, X is the case. This is to be contrasted with verbs like *claim* or *believe*: If Smith *believes* that there are ghosts in his attic, then we can by no means conclude there are ghosts in his attic. Alternately, if Bush *claims* that Iraq has WMD's, then it is not necessarily the case that Iraq does have WMD's. What separates (19) from these examples is that, in order for, e.g., (19a) to be true, the complement of the verb *acknowledge* must also be true.

The following example is similar:

- (20) Tim pretended that the pants were green. \rightsquigarrow The pants were not green.

Unlike (19), the veridicality of example (20) presupposes the *falsity* of the main verb's embedded complement. That is, we still draw a conclusion about the truth of the main verb's embedded complement clause, but it is the opposite conclusion from the verbs in (19). Verbs like *forget* or *know* that presuppose the truth of their complements are called FACTIVE. Verbs like *pretend* that presuppose the falsity of their complements are called COUNTERFACTIVE.

This behavior of factives and counterfatives is preserved under logical negation. Observe the following inferences:

- (21) a. Sally did not forget that computers crash sometimes. \rightsquigarrow Computers crash sometimes.
 b. Tim did not pretend that the pants were green. \rightsquigarrow The pants were not green.

Sentence (21a) is the sentential negation of (19b), and (21b) is the negation of (20). Notice that the presupposition-based inferences to be drawn remain the same when we negate the sentence. This is true of factives and counterfactives in general.

2.5.2 Implicatives

Implicativity is a phenomenon similar in many ways to factivity, as outlined above, but different in a number of important respects. Consider the following inferences:

- (22) a. John managed to fix the computer. \rightsquigarrow John fixed the computer.
 b. Jim succeeded in taming a rhino. \rightsquigarrow Jim tamed a rhino.

Such inferences are known as ENTAILMENTS, and the verbs carrying such entailments (e.g. *manage* and *succeed* in the above) are known as IMPLICATIVES (Karttunen, 1971). Implicative verbs differ from factive verbs in that they are sensitive to the polarity of the environment in which they occur; that is, an implicative will behave differently depending on whether it occurs in a positive environment (with an even number of negations applied to it) or a negative environment (with an odd number of negations). This is demonstrated readily by the inferences in (23):

- (23) a. John managed to fix the computer. \rightsquigarrow John fixed the computer.
 b. John didn't manage to fix the computer. \rightsquigarrow John did not fix the computer.
 c. George forgot to feed the wombats. \rightsquigarrow George did not feed the wombats.
 d. George didn't forget to feed the wombats. \rightsquigarrow George fed the wombats.

We conclude from the sentence in (23a), by implicativity, that John fixed the computer; if, however, the implicative verb *manage* is negated, then we draw the opposite conclusion. The same is true for (23cd). This dependence on local polarity is the most important way, for the purposes of calculating textual inferences, that implicatives differ from factives (remember, as illustrated in (21), that inferences to be drawn from factive verbs remain the same regardless of the environment the verb appears in).

Implicative constructions also carry with them certain presuppositions. For example, *manage to X* carries the presupposition that doing *X* is difficult somehow. Similarly, *bother to X*, another implicative, carries with it the presupposition that doing *X* is somehow not important (if *Jim didn't bother to turn off the lights*, then turning off the lights is somehow not important to Jim). These kinds of presuppositions are much more difficult to compute than those found associated with factives. Calculating the presuppositions of factives consists of saying that, in the context in which the factive appears, the verb's complement is to be considered either true or false, depending on the verb. With these sorts of verbs, the presupposition to be calculated is different for each verb, and, further, is much more subjective. Further, and more importantly, these

types of presuppositions are largely tangential to the task of a question-answering system, so we do not typically need to worry about them. For a system for calculating such presuppositions, however, see Karttunen and Peters (1979).

Implicatives, just like factives, fall naturally into classes, with members of a class behaving identically with respect to entailment-based inferences to be drawn. Since *manage* entails the truth of its complement clause when it occurs in a positive environment (as in (23a)) and the falsity of its complement clause when it occurs in a negative environment (as in (23b)), it is referred to as `impl_pp_nn` (both in Bridge’s lexicon and in this thesis). Similarly, since *forget* entails the falsity of its complement when occurring in a positive environment (as in (23c)) and the truth of its complement when occurring in a negative environment (as in (23d)), we mark it as `impl_pn_np`.

These are the two two-way implicative classes (i.e. classes of implicatives that carry entailments with respect to their embedded complement clauses when occurring in both negative and positive environments). There are also four classes of one-way implicatives that bear entailments only when appearing on one type of environment. These classes are illustrated in (24):

- (24) a. Shelley confirmed that the group disbanded. \rightsquigarrow The group disbanded.
 b. Shelley didn’t confirm that the group disbanded. \rightsquigarrow (no inference)
 c. Amy declined to take their money. \rightsquigarrow Amy did not take their money.
 d. Amy didn’t decline to take their money. \rightsquigarrow (no inference)
 e. Tim attempted to escape. \rightsquigarrow (no inference)
 f. Tim didn’t attempt to escape. \rightsquigarrow Tim did not escape.
 g. Ryan hesitated to ask for help. \rightsquigarrow (no inference)
 h. Ryan didn’t hesitate to ask for help. \rightsquigarrow Ryan did not ask for help.

The class `impl_pp`, in which we can conclude the truth of the verb’s complement clause when the verb occurs in a positive environment and nothing when the verb occurs in a negative one, is illustrated by (24ab). Similarly, (24cd) represent class `impl_pn`, (24ef) represent class `impl_nn`, and (24gh) represent class `impl_np`.

One important characteristic of implicatives is that they can be iterated – nested recursively within other implicatives – and retain their behavior. Consider the following:

- (25) a. Ray managed to forget to wash his hands. \rightsquigarrow Ray did not wash his hands.
 b. Teodor didn’t attempt to begin to write his paper. \rightsquigarrow Teodor did not write his paper.
 c. Molly managed to attempt to win the prize. \rightsquigarrow (no entailment).

For more details on this phenomenon, and for a formalization of the calculation of such implicativity-based inferences recursively, see (Nairn et al., 2006).

3 Paraphrases and Phrasal Implicatives

This section will outline, from a higher-level perspective, a few aspects of various classes of paraphrases and introduce a class of constructions known as phrasal implicatives. First, paraphrases are introduced, and various means of representing them internally in a system like **Bridge** are discussed. Next, phrasal implicatives are introduced and defined, and their representation in the lexicon that **Bridge** uses is discussed.

3.1 Paraphrases

Intuitively, two distinct sentences A and B are PARAPHRASES of each other if both A and B mean the same thing. That is, sentence A is true if and only if sentence B is true. The question of whether paraphrases truly exist is an open one (and is perhaps not answerable). For example, do the following sentences

- (26) a. Marten kissed Dora.
b. Dora was kissed by Marten.

mean exactly equivalent things? Are the sentences

- (27) a. Faye knew that Sven was happy.
b. Faye knew Sven was happy.

equivalent in meaning? This is debatable, but is outside of the scope of this thesis. In fact, the question is largely irrelevant: for the purposes of a logical Question-Answering system such as **Bridge**, it is useful to assume that distinct sentences can convey identical semantic content;⁷ for example, we would like (26b) to be a consequence of (26a) (and vice versa). Similarly, (27a) and (27b) should say the same thing. Thus, we assume that two sentences can indeed be paraphrases of each other, and we must determine how best to handle this phenomenon.

Many types of paraphrase already behave correctly in the question-answering task because of the way the XLE syntax and semantics handles them. For example, everything contained in the AKR of (26b) will also be in the AKR of (26a); the converse is also true (the semantics turns passive verbs into active ones). Thus, it will be the case that (26a) implies (26b) and (26b) implies (26a), so these two sentences are already represented internally effectively as paraphrases of each other. The same will be true of (27): the syntax represents (27a) and (27b) sufficiently similarly so as to yield identical AKRs for them.

There are, however, many classes of paraphrases relevant to the question-answering task that are equivalent in meaning not necessarily because of their syntactic structure, but because of the semantics of the words in the sentence. Consider, for example, the sentences

- (28) a. Hannah's health improved.
b. Hannah's health became better.

⁷Note that this assumption may have to be revisited for other tasks, such as text generation. One sentence may be more natural and stylistically acceptable than another equivalent way of saying the same thing.

These two sentences are, for the purposes of the **Bridge** system, paraphrases of each other ((28a) is true iff (28b) is true). However, whereas examples (26) and (27) are paraphrases because of syntactic differences that have negligible effect on the semantics of the sentences, (28a) and (28b) mean the same thing by virtue of the semantics of the phrases *improve* and *become better*, which are effectively synonymous. The example

- (29) a. Steve and Ellen are dating.
b. Steve and Ellen are an item.

is similar, as the phrases *are dating* and *are an item* are essentially synonymous. More specifically, we could say that *to be an item* is synonymous with *to be dating* iff the set of Agents of the VP has cardinality two. Note, however, that two sentences that are paraphrases because of the semantics of their component words need not be so because of synonymy. Consider:

- (30) a. Veronica is Marten's mother.
b. Marten is Veronica's son.

Sentences (30a) and (30b) are paraphrases of each other because of the semantics of the words *mother* and *son* – if *A* is male and *B* is female, *A* is *B*'s son iff *B* is *A*'s mother. Therefore, (30a) is true in just those cases where (30b) is true. Note that (30a) and (30b) are paraphrases of each other because of the meaning of the words contained therein. Contrast this with the following:

- (31) a. Penelope lives in Chicago.
b. Penelope lives in the most populous city in Illinois.

Sentence (31a) is true if and only if sentence (31b) is. However, this is so because of a fact about the world, not necessarily because of the language making up the sentences. Whereas we know the sentences in (30) to be paraphrases of each other because of language-based knowledge (in this case, the definitions of the words *mother* and *son*), we need world-based knowledge to conclude that (31a) and (31b) are paraphrases of each other (namely, which city in Illinois has the most people).

A system like **Bridge** that needs some degree of natural language understanding must be able to conclude that, for example, (28a) implies (28b) and vice versa. If two sentences mean the same thing, what is the best way to represent them so this equivalence in meaning emerges? There are three general paradigms to consider as solutions to the problem. Suppose *A* and *B* are paraphrases of each other.

1. Expand the representation of both *A* and *B*, when they occur, to note that one is a paraphrase of the other. Applying this technique to (30) would mean augmenting the AKRs of (30a) and (30b) so that the (30a) AKR has a fact stating that Marten is Veronica's son and the (30b) AKR has a fact stating that Veronica is Marten's mother. This approach would essentially give *A* and *B* identical AKRs, each containing the facts to be found in both *A* and *B*.

This approach leads to very large representations of paraphrases. In, for example, (30), this is not obvious: the AKRs for both sentences in the pair are augmented by about one fact each. However, if there are, say, fifteen or twenty different paraphrases, each saying the same thing, the impracticality of this approach becomes clear. For example, the sentences

- (32) a. Natasha and Amir are married.
b. Amir is Natasha's husband.
c. Amir married Natasha.
d. Amir and Natasha are husband and wife.

are just a fraction of a relatively large set of paraphrases of the same state of affairs. Thus, were we to follow this approach in representing a sentence from (32) so that all appropriate inferences can be drawn (which is, essentially, making the AKR for any of the sentences in (32) equal to the union of the AKRs of all paraphrases of that sentence), we would get very long resultant AKRs. This is suboptimal because longer AKRs take more space to store, make debugging more difficult, and trigger more rules (rewrite rules are introduced in §4.1).

2. A somewhat more reasonable way to represent paraphrases is to canonicalize – keep A 's representation the same, and map B 's representation to be equivalent to A 's. We are, in essence, picking out a canonical way to phrase something and mapping non-canonical paraphrases to the canonical phrasing. So, for example, we could choose to make the AKR of (32a) the canonical representation of the marriage relation and map the AKRs from (32bcd) to the AKR of (32a). This has the benefit of keeping the size of the AKR relatively minimal. Further, there is less work to be done in this approach as opposed to approach 1: the canonical form will be shorter than a representation containing the information from every paraphrase of the sentence.⁸ However, canonicalizing to a representation like (32a) has the negative consequence of removing the information that Amir is Natasha's husband and Natasha is Amir's wife. That is, in cases where it is ambiguous whether a name belongs to a male or female, we cannot infer who is the wife and who is the husband from a sentence like (32a) alone.
3. We could choose an independent third representation (call it C) to which we map both A and B . This could make sense in, for example, (33).

- (33) a. Winslow lost the race to Hannah.
b. Hannah won the race against Winslow.

Both of these sentences are paraphrases of each other. Winning and losing are converses of each other, and one occurs only in the presence of the other. Rather than canonicalizing

⁸In computational terms, if there are n ways to paraphrase a statement, when any of them is encountered, approach 1 will take $O(n)$ time, as it must add at least one fact for each of the n distinct paraphrases, whereas approach 2 will take $O(1)$ time, as mapping to a canonical form will take a constant amount of time, no matter how many ways to say it exist.

and giving precedence to, say, a representation like (33b), we could cast both (33a) and (33b) into an identical “contest event,” (or something like it) that has both a winner and a loser. This is a rational choice for such symmetrically behaving constructions. Consider the sentence:

(34) Winslow lost the race.

It is unspecified in (34) to whom Winslow lost the race; we know, however, that Winslow lost the race to *someone*. Thus, given the passage/query pair

- (35) a. Winslow lost the race.
b. Someone won the race against Winslow.

the **Bridge** system should say that (35a) follows from (35b) and vice versa (i.e. the two are paraphrases of each other). Mapping both to an independent third representation with a winner and a loser makes sense given that either the winner or the loser may not be specified.

All of the above approaches to representing paraphrases are similar. In each, we divide up the set of sentences in the language into equivalence classes, where each sentence in a class is a paraphrase of every other sentence in that class. The approaches given above can be viewed as different ways of picking out a unique representative for each class of paraphrases. Note that the representative of a class of paraphrases need not map exactly to a sentence *in* that class of paraphrases; in fact, approach 2 is the only approach given in which this will be the case.

3.2 Phrasal Implicatives

There is a large class of multiword constructions that are semantically similar to the implicative verbs introduced in §2.5.2. We call these constructions PHRASAL IMPLICATIVES. Consider the sentence

(36) Marcus took the time to tie his shoes properly.

From (36), we conclude that Marcus did, indeed, tie his shoes properly. The phrase *take the time* is a phrasal implicative. There are numerous types of phrases included in this class. There are adjectival phrasal implicatives such as *be able* (if *Marcus wasn't able to see the sun*, then we know Marcus did not see the sun) and *be too afraid to* (if *Marcus was too afraid to speak up*, then Marcus surely did not speak up). There are also, in addition to *take the time*, quite a few phrasal implicatives of type *verb NP to*, such as *have the time*, *see a way*, and *take the opportunity*. It is primarily implicative phrases of this latter type that will be outlined and categorized below.

As with single-word implicatives, phrasal implicatives fall into classes, depending on their behavior in environments of different polarity. There are `impl_pp_nn` phrasal implicatives:

- (37) a. Jason had the nerve to call her fat. \rightsquigarrow Jason called her fat.
b. Jason did not have the nerve to call her fat \rightsquigarrow Jason did not call her fat.

There are also `impl_pn_np` phrasal implicatives:

- (38) a. Quincy wasted the chance to surprise Paige. \rightsquigarrow Quincy did not surprise Paige.
b. Quincy didn't waste the chance to surprise Paige. \rightsquigarrow Quincy surprised Paige.

and `impl_nn` phrasal implicatives:

- (39) a. Roger didn't have the chance to go fishing. \rightsquigarrow Roger did not go fishing.
b. Roger had the chance to go fishing. \rightsquigarrow (no inference)

These implicative phrases behave similarly to single-word implicatives with respect to the inferences we can justifiably and systematically draw from their occurrence. For example, with one-way implicatives such as (39) above, we can only systematically conclude something from an occurrence of the phrase in one particular polarity environment. That is, we can conclude nothing valuable from (39b).⁹

3.2.1 Conditional Lexical Properties

Phrases like *take the time* or *have the nerve* only carry their implicativity when they appear intact. For example, the word *take*, by itself, is not implicative (e.g. if *Alex took the hat to ward off demons*, we do not know whether Alex did ward off demons or not); neither is the NP *the time* by itself implicative (e.g. if *Alex knew the time to go to the store*, we can conclude nothing about Alex going to the store). Also note the verb and NP constituting a particular phrasal implicative may carry different implicativity behavior when combining with a different NP or verb – for example, the phrases from (38) and (39) above behave differently, even though both share the NP *the chance*. Implicativity arising from phrases, then, arises only from the combination of a particular verb and a particular NP.

It is natural to represent such behavior in the system's lexicon. Since there appear to be a great many syntactically-similar phrasal implicatives of the form *verb NP to VP* (viz. *make the time to X*, *make the effort to X*), we make use of the following observation: a verb is conditionally implicative, depending on what its complement is. So, for example, if *have* takes the complement *the time*, then it is `impl_nn`. If, on the other hand, *have* takes the complement *the nerve*, then it is `impl_pp_nn`; if its complement is *the shoes* (e.g. *I had the shoes to dance*), then the phrase has no implicative behavior.¹⁰ Thus, the implicativity of a VP with respect to its embedded complement clause is dependent on the complement that the VP's head verb takes.

⁹Phrasal implicatives also carry with them presuppositions, much as single-word implicatives. For example, *take the time to X* carries with it the presupposition that doing *X* takes some amount of time. Similarly, *waste the opportunity to X*, another phrasal implicative, carries with it the presupposition that there was a time period in which doing *X* was possible, and that time period ended before the utterance of the statement. Just as before, not only are these presuppositions unsystematic, but they are relatively irrelevant for the purposes of a logical Question-Answering system.

¹⁰This is an important point, as one possible structural interpretation of the sentence *I had the shoes to dance* gives it the same structure as, e.g., the sentence *I had the cajones to dance*. We thus must make sure that the former sentence does not have the same implicativity behavior as the latter.

This conditional behavior is fairly systematic. These syntactically similar phrases fall naturally into groups with similar nouns and identical implicativity behavior. For example, *take the liberty*, *take the trouble*, and *take the initiative* all behave with identical implicativity. This fact is represented in the lexicon by having the word *take*, with the syntactic subcategorization frame V-SUBJ-OBJ, belong to the lexical class

```
conditional(impl_pp_nn, Theme, effort_noun).
```

That is, the word *take*, with subcat frame V-SUBJ-OBJ, behaves with implicativity characteristic of the class `impl_pp_nn` when its semantic Theme is of the class `effort_noun` (a somewhat poorly named class that includes the words *liberty*, *trouble*, and *initiative*). This results in the appropriate phrases having the appropriate implicativity behavior.

The conditional lexical properties of verbs appearing in phrasal implicatives of the class under consideration (those of the form *verb NP to VP*) can thus be represented succinctly in the system's lexicon. Not only must the verbs' lexical entries be augmented to note that they conditionally exhibit implicative behavior, but the nouns' lexical entries must also be augmented to note that they belong to the appropriate noun class. So, in the example above, the appropriate entry for *take* must be augmented to note that it belongs to the class `impl_pp_nn` when it takes a noun of the class `effort_noun` as a complement, and the appropriate lexical entry for all nouns of the class `effort_noun` must be augmented to include membership in this class. If these steps are both done, checking for implicativity in phrases consists of the relatively simple task of checking to see if a conditionally implicative verb combines appropriately with a noun of the correct class. The entry for *take* with the appropriate subcategorization frame contains the following facts:

```
(cat(V), word(take), subcat(V-SUBJ-OBJ),
 concept(%E),
 xfr:lex_class(%E,
 conditional(impl_pp_nn, Theme, chance_noun)),
 xfr:lex_class(%E,
 conditional(impl_pp_nn, Theme, asset_noun)),
 xfr:lex_class(%E,
 conditional(impl_pp_nn, Theme, effort_noun)),
 source(hand_annotated)).
```

which state that *take* is of class `impl_pp_nn` when it takes any of the three listed classes of nouns as Theme. The subcategorization frame for *trouble* will contain the following:

```
(cat(N), word(trouble), subcat(NOUN-XCOMP),
 concept(%E),
 source(hand_annotated),
 xfr:lex_class(%E, effort_noun)).
```

which states that it is of the class `effort_noun`. The behavior of phrasal implicatives can be thus completely encoded in the lexicon, with certain verbs marked as being conditionally implicative, depending on what complement they take.

Have	+	{	Ability Noun	(ability/means)	= impl_nn
			Chance Noun	(chance/opportunity)	= impl_nn
			Character Noun	(courage/nerve)	= impl_pp_nn
Take	+	{	Chance Noun	(chance/opportunity)	= impl_pp_nn
			Asset Noun	(money)	= impl_pp_nn
			Effort Noun	(trouble/initiative)	= impl_pp_nn
Use	+	{	Chance Noun	(chance/opportunity)	= impl_pp_nn
			Asset Noun	(money)	= impl_pp_nn
Waste	+	{	Chance Noun	(chance/opportunity)	= impl_pn_np
			Asset Noun	(money)	= impl_pp_nn
Miss, Lose	+	{	Chance Noun	(chance/opportunity)	= impl_pn_np
Seize, Grab	+	{	Chance Noun	(chance/opportunity)	= impl_pp_nn

Figure 7: Rubric of Implicativity

3.2.2 The Characterization of Verbs and Nouns

The regularity observed in phrasal implicatives of the form *verb NP to VP* is captured in Figure 7. For example, the word *have*, when it takes as a complement any member of the set of nouns marked in the lexicon as “ability noun” (any of *choice, energy, flexibility, heart, means, way, wherewithal, ability, freedom, will*), is of the class `impl_nn`. Note that there are phrasal implicatives of other syntactic forms that are not discussed as much in this thesis. Also, Figure 7 is not exhaustive with respect to phrasal implicatives of the form *verb NP to VP*; there are surely more such phrases in the language.

3.2.3 Implication or Implicature?

The question arises as to whether many of these entailments are *bona fide* logical inferences or, instead, mere CONVERSATIONAL IMPLICATURES. Consider, for example, the inferences

- (40) a. Linus didn’t attempt to close the door. \rightsquigarrow Linus did not close the door.
b. Lucy had the courage to ask for a raise. \rightsquigarrow Lucy asked for a raise.

Both of these are, according to the above analysis, valid inferences. We can, however, without contradiction, say something like

- (41) a. Linus didn’t attempt to close the door, but tripped on his way in and accidentally did so anyways with his foot.
b. Lucy had the courage to ask for a raise, but the boss was in Tahiti, so there was no opportunity.

It seems, then, since these inferences can be canceled without the introduction of any sort of contradiction, that such “entailments” are little more than systematically occurring conversational implicatures.

The most immediate response to this objection is a simple one – for the purposes of a Question-Answering system that processes English *as it is used*, the given analysis, performing all implicativity-based inferences, is acceptable. The implicativity judgments captured in Figure 7 are based not only upon intuition-based judgments, but also upon the use of these phrasal implicatives in naturally-occurring corpora. Searching for such a phrase’s use on Google, for example, will show that when the implicative phrases classified in Figure 7 occur naturally, the implications outlined in this thesis are, in the vast majority of cases, meant to be drawn. Speaking, then, from a pragmatic point of view, the drawing of the various inferences outlined in this thesis is justifiable for a system meant to answer questions about naturally occurring texts. For the purposes of natural language processing, it is not very important that these implicativity-based inferences be defeasible when these phrases are, in general, used precisely for their implicative properties. Note, however, that these inferences are drawn only in the presence of certain lexical markings, and if an application arises in which we wish to draw only strict, nondefeasible inferences, we could easily withhold such inferences by either modifying the lexicon or toggling off the processing of these lexical markings.

4 Implementation and Operation

This section will outline the implementation of the above phenomena in the **Bridge** system. §4.1 will outline the rewrite system used to implement the majority of higher-level processing in the **Bridge** system. §4.2 will give the code for processing phrasal implicatives and explain it. Finally, §4.3 will trace through this code operating on an example sentence.

4.1 The Rewrite System

4.1.1 Introduction to the Rewrite System

The system of rewrite rules is explained in depth in Crouch (2005) and Crouch et al. (1993-2007). The output of the XLE parser (a packed f-structure – cf. §2.2) can be represented entirely as a set of facts. This set of facts can be operated upon by certain rewrite rules. The mechanism of rewrite rules is sufficiently powerful for all of the processing to be done upon the representation of sentences. That is, the processing from a syntactic representation to a semantic representation and, further, from a semantic representation to AKR, can be performed entirely by applying a sequence of rewrite rules. This is an ordered rewrite system; that is, the rewrite rules are given in an ordering (R_1, R_2, \dots, R_n) , where rule R_1 is executed first, rule R_2 is executed after rule R_1 , and so on.

A rewrite rule is of the form $LHS \Rightarrow RHS$, where LHS and RHS are both sets of facts. It takes as input a set of facts. The LHS determines whether a rule applies to a given input or not. The RHS is a set of facts to be added to the output of the rule, and is added if and only if the LHS matches the input. A fact from the LHS of a rule will, in general, have two different types of atoms to match to a fact from the input to that rule: literals and variables. Variables are

prefaced with a % symbol; literals are not. They differ in that a literal must match exactly with part of the input, and a variable can match to anything. Thus, the following

```
context (t)
```

appearing in the LHS of a rule will match only to a fact to which it is exactly identical, character-by-character. It contains only literal terms, and will thus match only literally to an identical fact. So the above, if appearing on the LHS of a rule, will match to the fact `context (t)` and only this fact. However, the fact

```
context (%ctx)
```

will match the variable `%ctx` to any symbol. So this fact, appearing in the LHS of a rule, will match to `context (t)`, `context (ctx (go))`, and `context (XYZ)`, among others. Note it will not match to, e.g., `top_context (t)` (because the literal `context` is not equal to the literal `top_context`). It will not match either to `P (context (t))`. This is so because the outer literal in the above LHS fact, `context`, is not equal to the outer atom of the given fact, `P`. That the LHS fact matches to a portion of the given fact does not mean it matches the fact itself.

The LHS of a rule has to match to some subset of the input (which is, recall, a set of facts), and not necessarily the whole set of input facts. So a rule with LHS `context (t)` will “fire” on any input set of facts, no matter how large, if and only if it contains the fact `context (t)`. That is, the RHS of this rule will be added to a set of input facts, no matter how large, iff `context (t)` occurs in the input.

Whether a fact in the LHS is included in the output of a rule depends on whether it is prefaced by a + or not: if a fact in the LHS has a + in front of it, then the fact that matches to it in operation will be included in the output of a rule. If a fact appears in the LHS of a rule with no + in front of it, then any fact matching to it in operation of the rule will *not* be included in the output of the rule.

Note also that a fact can be prefaced by a “-” sign; this is something like logical negation, and is, perhaps unintuitively, unrelated to the + symbol. If the LHS of a rule is

```
-context (t)
```

then this rule will “fire” iff the fact `context (t)` does *not* appear in the input to the rule. So prefacing a fact in the LHS of a rule with a - symbol has the effect of reversing its effect: the rule fires only if there is *no* fact in the input that can be matched to this fact.

There are more important details of the rewrite rule system:

- We have at our disposal a constant `true` that matches any input. That is, a rule of the form

$$\text{true} \implies \varphi_1, \dots, \varphi_n$$

will add the facts $\varphi_1, \dots, \varphi_n$ to the output regardless of the input to this rule. If this rule takes a set Σ of facts as input, its output will be $\Sigma \cup \{\varphi_1, \dots, \varphi_n\}$.

- A rewrite rule can be optional. An optional rewrite will be of the form

LHS $\text{?}=>$ RHS.

Such a rule introduces a disjunction, where the output of the rule represents *either* the unmodified input of the rule *or* the the input with the rule applied. A disjunction like this corresponds to a possible split in the choice space; that is, it introduces some sort of ambiguity in representation (see §2.3 for more information about packed representations and choice space).

- The LHS of a rule need not be a single fact; we have propositional conjunction, disjunction, and negation handy:

- Conjunction is represented in the LHS of a rule by

φ, ψ

which is true iff φ and ψ are both true. That is, the rule

$\varphi, \psi \implies \chi$

will add χ to the output of the rule if φ and ψ can both be matched to facts in the input.

- Disjunction is represented in the LHS of a rule by

$(\varphi | \psi)$

which is true iff either φ or ψ is true.

- Negation, as noted above, is represented in the LHS of a rule by

$\neg\varphi$

which is true iff φ is false.

- A recursive rewrite rule of the form

LHS $\ast=>$ RHS

will apply to its own output until it can no longer be applied.

- External procedural calls can be included in the LHS of a rule by surrounding such a call with curly braces $\{ \}$. Such external procedure calls do not modify the set of input facts, but instead perform some sort of table lookup or condition test. For example,

$\{ \%A = \%B \}$

returns true iff the value of $\%A$ is the same as the value of $\%B$.

- One can define macros for the parameterization of commonly occurring patterns of rules. A call to such a macro consists of the @ symbol followed by that macro's name. This is a means of abstraction – it reduces redundancy in code and eliminates “copy and pasting” by allowing one to group together commonly used and meaningful patterns of rules into one macro.
- The RHS of a rule can consist of the empty set, written 0. A rule of the form

$$\varphi_1, \dots, \varphi_n \implies 0$$

where $\varphi_1, \dots, \varphi_n$ are facts, has the effect of removing any $\varphi_1, \dots, \varphi_n$ not prefaced by a + from the output set.

4.1.2 Simple Example Rewrite Rules

Consider the rewrite rule given in Figure 8. It is a simple rule that adds the fact on the RHS (with values in its variables) if both facts of the LHS are matched to facts in the input.

```
+role (Agent, %VerbSk, %AgentSk),
+role (Theme, %VerbSk, %ThemeSk)
==>
performedActOn (%VerbSk, %AgentSk, %ThemeSk)
```

Figure 8: Example Rewrite Rule

The semantics of this particular rule are not at all relevant to the present discussion; we are interested merely in its operation. Consider, then, the set of facts given in Figure 9, which we treat as potential input to the rule from Figure 8.

```
context (t),
role (Theme, punch:43, William:342),
role (Agent, punch:43, Jim:51)
```

Figure 9: Example Input to Rule

It is worth noting that this input does not correspond to a complete representation of any English sentence at any stage of processing, and is used here merely as an example to illustrate the operation of the rule in Figure 8. It is also worth noting, once more, that there is nothing inherently meaningful about the symbol names chosen. The LHS of this rule has a number of variables, and these variables must be assigned values in order to match the rule's input (which will never contain variables). The variable substitution¹¹ in Figure 10 will make the LHS match with a subset of the input to the rule from Figure 9.

¹¹A variable substitution is a finite mapping of variables to terms.

```

%VerbSk    ↦  punch:43
%AgentSk   ↦  Jim:51
%ThemeSk   ↦  William:342

```

Figure 10: Example Variable Substitution

That is, if one assigns to the three variables the values given by the mapping in Figure 10, then the LHS of the rule in Figure 8 will match to a subset of the input rules given in Figure 9. More specifically, the first conjunct of the LHS of the rule matches to the input fact

```
role (Agent, punch:43, Jim:51)
```

and the second to

```
role (Theme, punch:43, William:342)
```

with the given variable substitution. Thus, the rule will fire, and the RHS of the rule will be added to the output, which is given in Figure 11.

```

context (t),
role (Theme, punch:43, William:342),
role (Agent, punch:43, Jim:51),
performedActOn (punch:43, Jim:51, William:342)

```

Figure 11: Example Output of Rule

Since both of the conjuncts appearing in the LHS of the rule from Figure 8 are prefaced by + symbols, the facts from the input that they matched to are also included in the output. There is one fact in Figure 11 that did not occur in the rule’s input: the RHS of the rule in Figure 8 with the variable substitution from Figure 10 applied to it.

4.2 Rewrite Rule for Phrasal Implicatives

We now discuss the implementation of phrasal implicative processing in the **Bridge** system. More specifically, we discuss the processing of phrasal implicatives of the form *verb NP to VP*, which are all handled by a single rule. The brevity of the code involved is due to a number of factors. First of all, the system already had a fully implemented polarity-based calculus for implicativity-based entailments. That is, verbs like *manage* or *deign* were already marked in the lexicon as being implicatives, and the inferences described in §2.5.2 were already being drawn upon them. The phrasal implicative calculations can easily “piggyback” off of this pre-existing functionality. The second reason that this additional functionality could be implemented with such a relatively small amount of code is that phrasal implicatives behave very systematically, as instantiated by Figure 7. This systematicity is, as noted previously, encoded into system’s lexicon.

As noted in §3.2.1, the phrasal implicatives are stored in the lexicon as conditional combinations. That is, the lexical entry for, say, *miss* has a fact stating that when it takes any noun marked as a “chance noun” in the lexicon, it carries a certain implicativity signature. What the single

rewrite rule for handling phrasal implicatives of the form *verb NP to VP* does is to recognize and mark such occurrences. That is, the rule will (i) recognize a verb marked as conditionally implicative when followed by a certain noun, e.g. *miss*; (ii) recognize such an appropriate noun, e.g. *chance*, and (iii) mark that verb phrase (in this case, *miss the chance*) as implicative (with the appropriate implicativity signature, in this case, *impl_pn_np*) in this occurrence.

```

1  @is_lex_class(%VerbSk, conditional(%class, %Role, %NounClass)),
2  +role(%Role, %VerbSk, %NounSk),
3  @is_lex_class(%NounSk, %NounClass),
4  ( +context_relation(%HostCtx, %CompCtx, crel(sem_xcomp,%NounSk)) |
      +context_relation(%HostCtx, %CompCtx, crel(Predicate, %VerbSk))),
5  +context_head(%CompCtx, %CompSk),
6  +role(Agent, %VerbSk, %SubjSk),
7  +temporalRel(%TempRel, %T2, %VerbSk),
8  role(%CompRole, %CompSk, null_pro:%%)
   ==>
9  context_relation(%HostCtx, %CompCtx, crel(sem_xcomp,%VerbSk)),
10 lex_class(%VerbSk,%class),
11 temporalRel(%TempRel, %T2, %CompSk),
12 role(%CompRole, %CompSk, %SubjSk).

```

Figure 12: Rewrite rule for Phrasal Implicatives of form *verb NP to VP*

The implementation of the rule is shown in Figure 12. Lines have been numbered for ease of reference. The rule is terse – relatively concise for the amount it does. Line 1 checks to see if the sentence being processed contains a verb marked in the lexicon as conditionally implicative, dependent upon what noun appears as its complement. Remember that, since `@is_lex_class` is prefaced with a `@` symbol, this is a macro, written elsewhere; the details of this macro’s implementation are not important. This line will be obtain iff the sentence contains a verb (whose Skolem constant will be assigned to the variable `%VerbSk`) that is marked in the lexicon as having a conditional property. The parameters of this conditional property will be put in the variables `%class` (corresponding to something like `impl_pp_nn`), `%Role` (corresponding to something like `Theme` – the semantic role, ultimately from `VerbNet`), and `%NounClass` (corresponding to something like `effort_noun` or `chance_noun`).

Line 3 checks to see if there is a noun appearing in the sentence that is of the appropriate lexical class so as to trigger the conditional implicativity of the verb picked out by line 1 of the rule (for example, if the verb *miss* appears, the third line would check to see if some noun of type `chance_noun` appears in the sentence). Remember that the appropriate noun class must, according to line 1, be stored in the variable `%NounClass`; line 3 is checking to see if there is such a noun in the sentence. If so, its Skolem constant will be stored in the variable `%NounSk`.

Note, however, that these two lines of code alone (the first and third) will positively mark sentences like *John missed Las Vegas’s many games of chance*, because this sentence contains both the verb *miss* (and, indeed, it appears with the same subcategorization frame, `V-SUBJ-OBJ`, as *miss* in *John missed the chance to say goodbye*) and the noun *chance* (which is marked as

being of type `chance_noun`). We therefore need line 2 to ensure that the verb and noun picked out by lines 1 and 3 are related in the appropriate way (namely, that the noun plays the semantic role for the verb as required by the lexical entry for the verb, which will be stored in the variable `%Role`). So, in this example, if *chance* is not playing the role of Theme to the verb *miss* (as in the sentence *John missed the chance to say goodbye*), then the phrase will not be processed as implicative.

Lines 4 and 5 pick out the contextual structure for this part of the sentence (that is, they get the names of the outer context and the context of the phrase's embedded complement and store them in the variables `%HostCtx` and `%CompCtx`). Note that line 4 is a disjunction of two possible terms, written (term1 | term2). This expression will match if either of its disjuncts match against some term in the input. This is present so as to match appropriately for passive sentences such as *Time was taken to finish*, which are represented slightly differently. In a sentence like *John took the time to eat*, there will be three contexts – the outer context τ , a context representing the complement of *take* (namely, *the time*), and the context of the embedded phrase (in this case, *to eat*). Lines 4 and 5 make sure these contexts are present and properly related (that is, that the input sentence is of the correct form), and pick out the first and third contexts (the outer context and the inner context) and assign their values to variables.

Lines 6 and 7 anchor facts needed for adding the necessary information to the output of the rule. Line 6 gets the name of the Skolem constant representing the Agent of the verb (this will be the constant for *John* in *John missed the chance to say goodbye*). This is stored in the variable `%SubjSk`. Line 7 gets the temporal relation of the verb (that is, if *John took the time to X*, then *X* happened in the past, whereas if *John will take the time to X*, then *X* will happen in the future). This temporal relation is matched to the variable `%TempRel`. These two bits of information are useful because we want the output of this rule to note that if *John took the opportunity to say hello*, then it is John who said hello (the semantics output of XLE does not carry this information). Further, the sentence's representation should also note that John said hello in the past (again, the XLE semantics note only that *take* is past tense, and not that the act of saying hello also happened in the past). In order to add these two facts to the output of the rule, we must first get the Agent of the implicative phrase and the tense of this implicative phrase.

Finally in line 8, the semantic role that the main verb of the embedded complement's subject takes is found (this is, for example, the role that the lexicon says the subject of *say* should take in the sentence *John missed the chance to say goodbye*). This value is stored in the variable `%CompRole`. This information will also be used to form the rule's output. Note this rule is not prefaced with a `+`: there is no reason to keep this fact around, so it is deleted.

On the RHS of the rule are just four facts. Line 9 “hooks up” the contextual structures correctly so that the already-present implicativity functionality will work correctly. It adds a fact stating that the outer context and the inner context (the ones matched to `%HostCtx` and `%CompCtx`) are related by the main verb of the implicative phrase (e.g. *take* in *take the time*). Line 10 performs the important job of marking that the implicative verb phrase (e.g. *miss the chance*) is, indeed, implicative in this occurrence. With these two tasks done, the implicative calculus already present in the system will be triggered and will add the appropriate inference to the AKR.

Line 11 adds a fact stating that the complement of the implicative verb phrase takes that implicative verb's temporal tense. For example, if the sentence is *John took the time to sleep*, then this line adds a term stating that the sleeping act occurred in the past. Finally, and crucially, line 12 adds a term to the output stating that the subject of the verb occurring as complement to the phrasal implicative is the subject of the phrasal implicative itself. So, for example, this term will note that, if *John took the time to sleep*, then John was the one who slept.

This one rule, then, handles all phrasal constructions of the form *verb NP to VP*, appropriately modifying the intermediate AKR so that, when input to the already-present implicativity processing, the desired inferences are drawn. It bears repeating that the implicativity behavior of these syntactically similar phrases is systematic, and it is thus most natural to store the information contained in Figure 7 (namely, which combinations of verbs and noun phrases behave implicatively) in the lexicon. This rewrite rule recognizes an appropriate combination of these lexically annotated words and performs manipulations so that the correct inferences are drawn.

4.3 Examples of Operation

Take the sentence *Maura did not take the time to learn Arabic*. The input to the rule given in §4.2 when processing this sentence as a passage has 77 facts. The majority of these facts are not relevant to the rule. The relevant facts given as input to the rewrite rule from §4.2 are given in Figure 13.

```

1. lex_class (take:2, conditional (impl_pp_nn, Theme, asset_noun))
2. role (Theme, take:2, time:5)
3. lex_class (time:5, asset_noun)
4. context_relation (ctx (take:2), ctx (learn:8),
                    crel (sem_xcomp, time:5))
5. context_head (ctx (learn:8), learn:8)
6. role (Agent, take:2, Maura:n (0, 1))
7. temporalRel (startsAfterEndingOf, Now, take:2)
8. role (Agent, learn:8, null_pro:6)

```

Figure 13: Relevant input to rewrite rule for sentence *Maura did not take the time to learn Arabic*.

The rule will operate as outlined in §4.2. The line numbers given in Figure 13 correspond to the line numbers matched to in Figure 12. That is, line 1 of Figure 13 matches line 1 of Figure 12 if we assign the value `take:2` to the variable `%VerbSk`, the value `impl_pp_nn` to the variable `%class`, and so on. The full variable substitution needed to unify (that is, make syntactically identical) the set of atomic terms in Figure 13 and the LHS of the rule given in Figure 12 is given in Figure 14.

Given this variable substitution, the input given in Figure 13 can be unified with the LHS of the rule from Figure 12. Thus, the RHS of this rule (with the substitution from Figure 14 applied) is added to the set of facts to be the output of this rule. These facts are given in Figure 15.

```

%VerbSk      ↪  take:2
%class       ↪  impl_pp_nn
%Role        ↪  Theme
%NounClass   ↪  asset_noun
%NounSk      ↪  time:5
%HostCtx     ↪  ctx(take:2)
%CompCtx     ↪  ctx(learn:8)
%CompSk      ↪  learn:8
%SubjSk      ↪  Maura:n(0,1)
%TempRel     ↪  startsAfterEndingOf
%T2          ↪  Now
%CompRole    ↪  Agent

```

Figure 14: Variable substitution for unification of rule LHS and rule input for example sentence

```

9.  context_relation(ctx(take:2), ctx(learn:8),
      crel(sem_xcomp, take:2))
10. lex_class(take:2, impl_pp_nn)
11. temporalRel(startsAfterEndingOf, Now, learn:8)
12. role(Agent, learn:8, Maura:n(0,1))

```

Figure 15: Relevant output from the rewrite rule

As in Figure 13, the lines are numbered to correspond to the lines in Figure 12 from which they were generated. The total input to the rule for this sentence was 77 lines. The output will be 80 lines. It will contain all facts not matched to by this rule (of which there will be $77 - 8 = 69$, as there are 8 lines on the LHS of the rule, each matching to a single fact) plus 7 (because 7 of the 8 lines on the LHS of the rule are prefaced by a +, which means they are added back to the output of the rule) plus 4 (because there are 4 lines in the RHS of the rule). The output of this rule, when fed as input to the existing calculus for single-word implicatives, will produce the appropriate inferences.

The given rewrite rule can also handle more complicated examples. For example, the passage/query pair depicted in Figure 1, which is

(42) P. John managed to miss the opportunity to score.

Q. John scored.

will give the answer NO. The phrasal implicative *miss the opportunity* is processed by the above procedure to yield the conclusion that, within the complement context of the verb *manage*, the scoring event did not happen. Since *manage* is *impl_pp_nn*, this is then raised to the outer context, and the AKR for (42P) thus contains a fact stating that the scoring event with John as its Agent did not happen. The appropriate inference can be drawn from this information. We have demonstrated, then, how phrasal implicatives of the form *verb NP to VP* are processed with a

single rewrite rule in conjunction with appropriate markings in lexical entries and the existing calculus for implicativity.

5 Conclusions and Possibilities for Further Research

Drawing inferences like those induced by phrasal implicatives is necessary for the task of entailment and contradiction detection, which is itself central to the question-answering task that *Bridge* performs. Implicative phrases divide up naturally into different classes, as instanced by Figure 7, and this systematic behavior lends itself well to representation within the lexicon. The actual processing of phrasal implicatives becomes relatively straightforward once they are properly encoded into the lexicon. The main challenge for a broad-coverage question-answering system, then, lies in properly identifying and classifying the many different types of implicative phrases present in English.

The construction *verb NP to VP* is only one class of phrasal implicatives. There are others. For example, verbs can take adjectival complements, as in:

(43) Jimmy made sure to call his mother this morning.

This sentence implies that Jimmy called his mother; the construction *make sure to VP* is implicative, but is not of the form *verb NP to VP*, which the phrasal implicatives discussed throughout this thesis are. Similarly, the sentence

(44) Rusty was too humble to accept his birthday present.

implies that Rusty did not accept his birthday present. The construction *be too ADJ to VP* is of the class `impl_pn`. Note, however, that

(45) Rusty was all too eager to accept his birthday present.

implies that Rusty *did* accept his birthday present. It appears, then, that the construction *be all too ADJ to VP* may be of the class `impl_pp`, unlike the similar construction *be too ADJ to VP*.¹² Consider, then,

(46) Jimmy was all too willing to ignore the cop's order.

It is not entirely evident in (46) whether Jimmy followed the cop's order or not – for example, if the cop did not give orders, then it could not be said that he ignored anything. Constructions of the form *all to ADJ to VP*, then, may or may not act implicatively, possibly depending on which adjective occurs in the phrase.

The list of phrasal implicatives given in this thesis (contained largely in Figure 7) is thus far from exhaustive (it does not exhaust the implicative phrases of the form *verb NP to VP*, for

¹²It may appear that this analysis is invalidated by the sentence *We were all too tired to listen to Quimby talk*. The phrase *be all to ADJ to VP* in this sentence, unlike in (45), implies that we *did not* listen to Quimby talk, and the construction, in this case, appears to be `impl_pn`. This is misleading, however, as the phrase “all too tired” in this sentence occurs differently, grammatically, from “all too eager” in (45) (*all* is a different part of speech in each).

example). More work can be done fleshing out the classes of implicative phrases in the English language.

If these other classes of phrasal implicatives prove to be as systematic as the class *verb NP to VP*, they they could each be handled in a system like **Bridge** by one rule apiece, similar to the one in Figure 12. Once the space of phrasal implicatives is better mapped out, the systematic behavior of these phrases will make the derivation of the appropriate inferences fairly straightforward.

Similarly, various classes of hereto unconsidered paraphrases will arise and will have to be considered by question-answering systems. One of the more widely applicable concepts outlined in this thesis is that of encoding conditional behavior exhibited by verbs lexically. This could potentially be applied to many phrasal constructions in which phrases exhibit behavior that their constituent words, in isolation, do not.

References

- Bobrow, Daniel G. Cheslow, Bob, Condoravdi, Cleo, Karttunen, Lauri, King, Tracy Holloway, Nairn, Rowan, de Paiva, Valeria, Price, Charlotte and Zaenen, Annie. 2007. PARC's Bridge and Question Answering System. In Tracy Holloway King and Emily M. Bender (eds.), *Proceedings of the Grammar Engineering Across Frameworks (GEAF) 2007 Workshop*, CSLI Publications, <http://csli-publications.stanford.edu/GEAF/2007/papers/geaf07bobrowetal.pdf>.
- Bobrow, Daniel G., Condoravdi, Cleo, Crouch, Richard, Kaplan, Ron, Karttunen, Lauri, King, Tracy Holloway, de Paiva, Valeria and Zaenen, Annie. 2005. A Basic Logic for Textual Inference. In *Proceedings of the AAAI Workshop on Inference for Textual Question Answering*.
- Condoravdi, Cleo, Crouch, Richard, van der Berg, Martin, Everett, John O., Stolle, Reinhard, de Paiva, Valeria and Bobrow, Daniel G. 2001. Preventing Existence. In *Proceedings of the Conference on Formal Ontologies in Information Systems*.
- Crouch, D. and King, T.H. 2006. Semantics via F-Structure Rewriting. In *Proceedings of LFG06*, pages 145–165, CSLI On-line publications, accessible at <http://csli-publications.stanford.edu/LFG/11/lfg06crouchking.pdf>.
- Crouch, Dick, Dalrymple, Mary, Kaplan, Ron, King, Tracy, Maxwell, John and Newman, Paula. 1993-2007. *XLE Documentation*. 1993-2001: Xerox Corporation, 2002-2007: Palo Alto Research Center (PARC).
- Crouch, Richard. 2005. Packed Rewriting for Mapping Semantics to KR. In *Proceedings Sixth International Workshop on Computational Semantics*.
- Crouch, Richard and King, Tracy Holloway. 2005. Unifying Lexical Resources. In *Proceedings of the Interdisciplinary Workshop on the Identification and Representation of Verb Features and Verb Classes*.

- Dagan, Ido, Glickman, Oren and Magnini, Bernardo. 2006. The PASCAL Recognising Textual Entailment Challenge. In J. Quionero-Candela, I. Dagan, B. Magnini and F. d'Alché Buc (eds.), *Lecture Notes in Computer Science*, volume 3944, Springer.
- Dalrymple, Mary. 2001. Lexical Functional Grammar. *Syntax and Semantics* 34.
- de Paiva, Valeria, Bobrow, Daniel G., Condoravdi, Cleo, Crouch, Richard, Kaplan, Ron, Karttunen, Lauri, King, Tracy Holloway, Nairn, Rowan and Zaenen, Annie. 2007. Textual inference logic: Take two. In *Proceedings of the Workshop on Contexts and Ontologies: Representation and Reasoning, Workshop associated with the 6th International Conference on Modeling and Using Context*.
- Fellbaum, Christiane (ed.). 1998. *WordNet: An Electronic Lexical Database*. The MIT Press.
- Kaplan, Ronald M. and Bresnan, Joan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In Joan Bresnan (ed.), *The Mental Representation of Grammatical Relations*, pages 173–281, MIT Press, reprinted in Dalrymple, Kaplan, Maxwell, and Zaenen, eds., *Formal Issues in Lexical-Functional Grammar*, 29–130. Stanford: Center for the Study of Language and Information. 1995.
- Karttunen, Lauri. 1971. Implicative Verbs. *Language* 47(2), 340–358.
- Karttunen, Lauri and Peters, Stanley. 1979. Conventional Implicature. *Syntax and Semantics* 11, 1–56.
- Kipper, Karin, Dang, Hoa Trang and Palmer, Martha. 2000. Class-based Construction of a Verb Lexicon. In *AAAI-2000 17th National Conference on Artificial Intelligence*.
- Levin, Beth. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press.
- Maxwell, John and Kaplan, Ron. 1991. A Method for Disjunctive Constraint Satisfaction. In Masaru Tomita (ed.), *Current Issues in Parsing Technologies*, pages 173–190, Kluwer.
- McCarthy, John. 1993. Notes on Formalizing Context. In *Proceedings of the 13th Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 555–560.
- Montague, Richard. 1970. The proper treatment of quantification in ordinary English. In Richmond Thomason (ed.), *Formal Philosophy. Selected papers of Richard Montague*, pages 247–270, New Haven 1974: Yale university Press.
- Nairn, Rowan, Condoravdi, Cleo and Karttunen, Lauri. 2006. Computing Relative Polarity for Textual Inference. In *Proceedings of ICoS-5 (Inference in Computational Semantics)*.