Technical Report

UCAM-CL-TR-182
ISSN 1476-2986

Number 182

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

The theory and implementation of a bidirectional question answering system

John M. Levine, Lee Fedder

October 1989

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

The Theory and Implementation of a Bidirectional Question Answering System

John M. Levine (jml@uk.ac.cam.cl)

Lee Fedder (lf@uk.ac.cam.cl)

University of Cambridge, Computer Laboratory,

Pembroke Street, Cambridge CB2 3QG, England.

Topic areas: bidirectional grammars, strategic and tactical generation.

This paper describes a question answering system which is a limited instance of the general bidirectional architecture suggested by Appelt (1987). The novel features of our approach include the use of a linguistically well-motivated set of functional features; a bidirectional grammar which encodes these features directly; a question answering program which uses the thematic organisation of the user's input to construct a cooperative reply; and a tactical generation component which can be used with Montague semantics.

1. Introduction

One of the main areas of research in computational linguistics is the construction of natural language dialogue systems. This paper describes our investigation into the use of a particular architecture in the construction of such a system. We begin by describing the general architecture under investigation, followed by an overview of our implemented system. The section concludes with a guide to the rest of the paper.

1.1 The General Bidirectional Architecture

The general bidirectional architecture is shown in Figure 1. The processing is divided into three distinct phases. Phase one (grammar-based interpretation) takes the input sentences of the user and computes an intermediate representation of those sentences using the syntactic and semantic information contained in the grammar. Phase two uses the information contained in the output from the first phase in conjunction with the information contained in the knowledge base to generate an intermediate representation

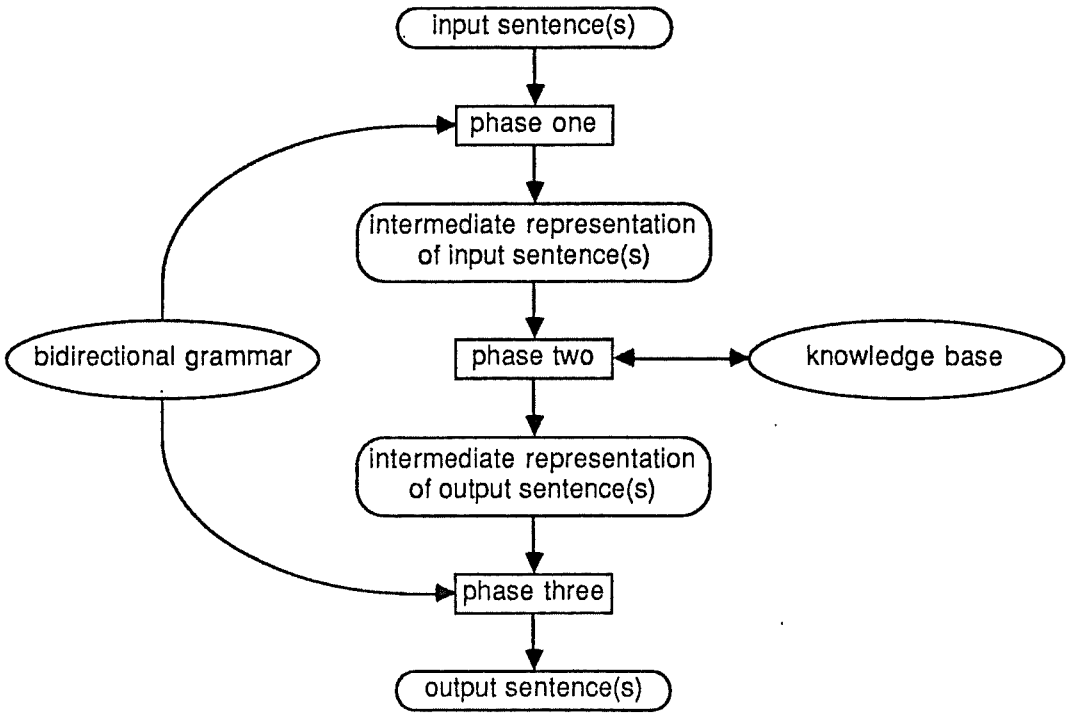

Figure 1 The General Bidirectional Architecture

of the system's response. For a general dialogue system, this phase might consist of anaphora resolution, plan recognition, rational interaction (deciding what plan the system has with regard to the user's plan), and strategic generation (determining the propositional content and functional organisation of the system's response). The same formalism is used for the intermediate representation of the input and the intermediate representation of the output. Phase three (tactical generation) uses the output from phase two in conjunction with the same grammar used by phase one to generate the natural language form of the system's response.

The use of this organisation for linguistic processing is proposed because it places emphasis on two desirable system characteristics, namely bidirectionality and modularisation. Bidirectionality implies that interpretation and generation can share the same sources of knowledge. Modularisation is desirable because it allows a system to be constructed from logically distinct parts. The proposed architecture makes a strong distinction between syntax and semantics (phases one and three) and pragmatics (phase two) in terms of the knowledge sources used during the processing. Levinson (1983: 21ff) gives a good discussion of the theoretical issues involved in making this distinction.

1.2 The Implemented System

The research completed so far has concentrated on the use of the general bidirectional architecture for the construction of a complete natural language question answering system. Our efforts have been directed towards the development of an appropriate formalism for the intermediate representation of sentences; the use of this formalism for strategic and tactical generation; and the implementation and use of a bidirectional grammar.

Two simplifications have been made to the general architecture. Firstly, since we wished to direct our efforts towards the generative capabilities of the system, we assume that the output of the grammar-based interpretation process encodes the full meaning of the user's input. Secondly, we have concentrated on the treatment of single question/answer pairs, without regard to discourse modelling. We hope to correct both

of these deficiencies in future versions of the system.

The organisation of the implemented system is shown in Figure 2. The Grammar Development Environment developed at Cambridge, Lancaster and Edinburgh (Briscoe et al., 1987; Carroll et al., 1988) was used to define the bidirectional unification grammar and also acts as the grammar-based interpretation component of the system. The basic syntactic analyses used are based on insights from contemporary theories such as GPSG (Gazdar et al., 1985) and LFG (Bresnan, 1982), and have been extended to cover a wider range of thematic forms. A simplified version of Montague semantics, similar to that used by Rosenschein and Shieber (1982), is used to compute the logical forms of sentences from their parse trees.

The formalism used for the intermediate representation of sentences is based on the use of standard first-order logical forms. However, the logical representation of a sentence does not reflect its functionality, which is needed for generating appropriate responses (Kaplan, 1983: 171). The proposed solution to this problem is to annotate the logical form with a number of functional features which are used during tactical

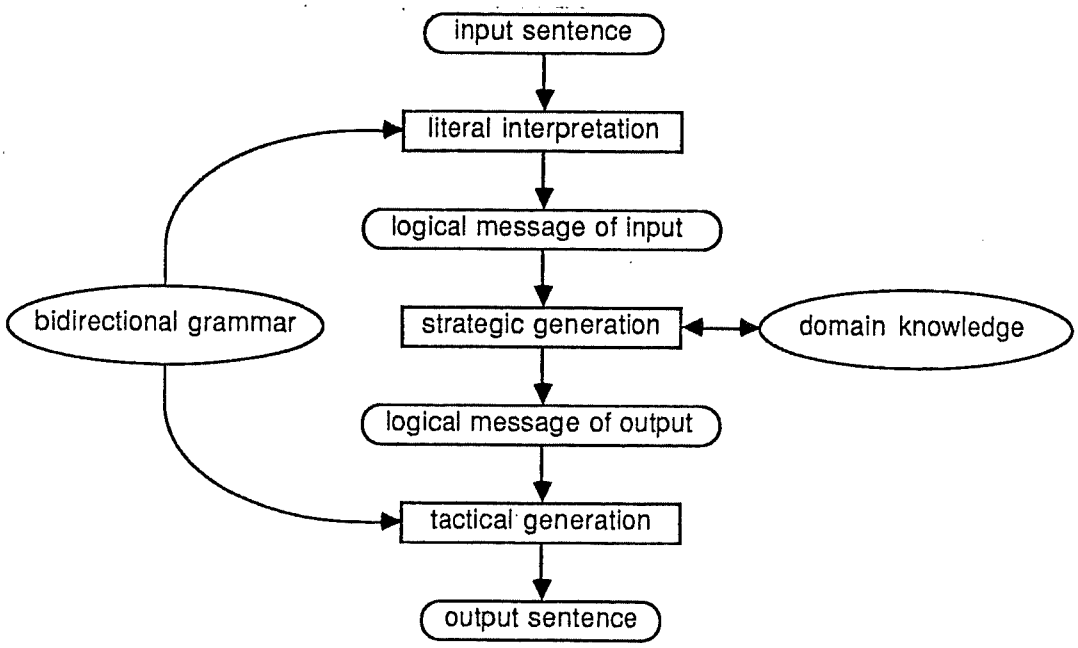

Figure 2 The Organisation of the System

generation, and constrain the grammar to provide a one-to-one mapping between unambiguous sentences and meaning representations. Considered as a single unit, the logical form and functional features for a particular sentence are known as the *logical message* of that sentence.

The system is written in Common Lisp and runs on a Hewlett Packard 9000 Series workstation under the HP-UX operating system. On average, a dialogue consisting of one question and answer pair takes 3.2 seconds of CPU time to run to completion. A sample dialogue with the system using a grammar that deals with statements and questions about Cambridge colleges is shown in Figure 3.

1.3 A Guide to Remaining Sections

The next two sections present the theory on which the system is based: Section 2 describes the set of functional features used in the system, and Section 3 discusses the use of a bidirectional grammar. The following three sections then describe the

---

```
Q: Which colleges have a chapel?
A: King's and Trinity have a chapel.
Q: When is King's chapel open?
A: It's open today.
Q: Is Trinity's chapel open today?
A: No, it's open on Wednesday and Saturday.
Q: Does Trinity have a library?
A: Yes, it does.
Q: Who designed Trinity's library?
A: It was designed by Wren.
Q: Was King's founded by Wren?
A: No, it was founded by Henry-VI.
Q: Did Wren found Trinity?
A: No, it was Henry-VIII who founded Trinity.
Q: When was King's founded?
A: It was founded in 1441.
```

Figure 3 A Sample Dialogue with the System

implementation: Section 4 describes the grammar used in the system, Section 5 describes the strategic generator and Section 6 the tactical generator. Section 7 presents a worked example of the entire system in operation, and the final section discusses some directions for further research.

2. Functional Features

As was mentioned in the previous section, the logical form of a sentence alone does not provide sufficient information to work from in the context of strategic and tactical generation. For example, the logical form

$$\exists x,e.\text{give}(\text{john},\text{mary},x,e) \wedge \text{book}(x) \wedge \text{past}(e)$$

could correspond to many different sentences, some of which are shown below.

(a) John gave a book to Mary.
(b) John gave Mary a book.
(c) Mary was given a book by John.
(d) It was John who gave a book to Mary.
(e) Did John give Mary a book?
(f) Was Mary given a book by John?
(g) Was it John who gave a book to Mary?
(h) Which book did John give to Mary?

The values of the functional features are governed by syntax and convey information about the communicative function of a sentence. Five functional features are used in the question answering system: sentence type, tense, theme, linguistic focus, and emphasis. These can be split into two groups: the sentence type and tense features convey information about the communicative status of the sentence as a whole, whereas the theme, linguistic focus, and emphasis features convey information about the communicative dynamism (Quirk et al., 1985: 1356) and structural organisation of particular elements of the linguistic message.

2.1 Sentence Type

The sentence type functional feature operates on the sentence as a whole and determines whether it is a declarative sentence, a yes/no question or a wh-question. This feature effectively encodes the surface speech act of the sentence (Appelt, 1985). Some examples of this are given below.

(a) John gave a book to Mary. [sentence type = declarative]
(b) Did John give Mary a book? [sentence type = yes/no question]
(c) Which book did John give to Mary? [sentence type = wh-question]

2.2 Tense

The tense feature specifies when the "action" described by the logical form takes place. Strictly speaking, the tense of a sentence is part of its truth-conditional semantics and should therefore be reflected in the logical form. We encoded it as a feature in our system because it reduces the search space required by the tactical generator in some circumstances. As long as the tense feature of the top sentence node gives us the tense of the sentence as a whole, this seems to be a reasonable treatment.

2.3 Theme

The theme of a sentence is formally defined by Quirk et al. (1985: 1361) as "the initial part of any structure when we consider it from an informational point of view." Similar definitions are given by Halliday (1967: 212) and Brown and Yule (1983: 126). Halliday points out that the theme of a sentence can be determined without regard to the context of the surrounding discourse; this means that the theme of a sentence can be determined solely in terms of syntax. Some examples of the use of this feature are given below.

(a) John gave a book to Mary. [theme = john]
(b) Mary was given a book by John. [theme = mary]
(c) It was founded by Henry-VI. [theme = it]

In terms of communicative function, the theme of a sentence constitutes what Halliday calls "a point of departure"; in the case of unmarked sentences, such as the ones above, the theme is the element of least communicative dynamism, and therefore it

is the one which is most readily pronominalised if it is contextually bound (i.e. given), as in (c).

In the case of interrogative sentences, we use a slight variation on the definition given above. The theme of an interrogative sentence is defined to be the initial element of the "declarative version" of the question. For example, the declarative version of "Was King's founded by Wren?" is "King's was founded by Wren." Some examples of this are given below.

(a) Did John give a book to Mary? [theme = john]
(b) Was Mary given a book by John? [theme = mary]
(c) Who was King's founded by? [theme = kings]

Defining the theme of interrogatives in this way gives more information to the strategic generator than defining the theme to be the interrogative word. For example, it allows the response "No, he gave a book to Jane" to be generated in response to (a). This process will be discussed in more detail in Section 5.

The linguistics literature contains references to a number of phenomena which are similar to our notion of theme. McKeown (1985: 75ff) gives a good overview of this work. Indeed, it seems that our use of the term "theme" is virtually synonymous with her use of the term "immediate focus."

2.4 Linguistic Focus

The linguistic focus of a sentence is defined as the contextually non-bound (i.e. new) portion of the linguistic message. Again, this definition is taken from the theoretical linguistics literature (Hajicova, 1983, 1987; Hajicova and Sgall, 1985; Quirk et al., 1985). Unlike the theme, we cannot directly define the linguistic focus in terms of syntax, since it is defined in terms of the context of the preceding discourse. However, using observations from the research referenced above, it is possible to reach a simplified definition of linguistic focus in terms of syntax alone. For example, in an unmarked sentence with a transitive or ditransitive verb, the linguistic focus is assumed to reside on the final noun phrase, by the principle of end-focus (Quirk et al., 1985: 1357). This

states that it is common to present the information in a message so as to achieve a linear presentation from low to high information value. Some examples of this are given below.

(a) He gave a book to Mary. [focus = mary]
(b) He gave Mary a book. [focus = $\exists x.book(x)$]
(c) Did John give a book to Mary? [focus = mary]
(d) Was Mary given a book by John? [focus = john]

In all these cases, the linguistic focus is the element of greatest communicative dynamism. In the case of declarative sentences like (a) and (b), the linguistic focus is used to specify the new information; for example, given the question "Who did John give a book to?" (a) is more appropriate as a response than (b). In the case of yes/no questions such as (c) and (d), the linguistic focus seems to indicate what the speaker is unsure about, and this can be used as a guide to the construction of an appropriate database query if the yes/no question receives a negative answer. For example, the response "No, he gave a book to Jane" is an appropriate response to (c) but not to (d).

2.5 Emphasis

The final functional feature used in the system is emphasis. This is a boolean flag which is set to true when special emphasis is applied to the linguistic focus of a sentence by the use of an it-cleft. Some examples of this are given below.

(a) It was John who gave a book to Mary. [focus = john, emphasis = true]
(b) Mary was given a book by John. [focus = john, emphasis = false]
(c) Was it Wren who founded King's? [focus = wren, emphasis = true]
(d) Was King's founded by Wren? [focus = wren, emphasis = false]

The use of this feature indicates that the degree of communicative dynamism of the linguistic focus is higher than that associated with the linguistic focus in unmarked sentences. In the case of declarative sentences such as (a), not only is the linguistic focus new, but it is unexpectedly new. In the case of yes/no questions, such as (c), it indicates that the speaker is especially unsure of a particular piece of information; indeed, (c) is not far removed from the wh-question "Who was King's founded by?" in its

communicative function.

3. Using a Bidirectional Grammar

The use of a single bidirectional grammar for both literal interpretation and tactical generation is an important idea, as noted by a number of leading researchers (Appelt, 1987; Jacobs, 1988; Shieber, 1988). There are many reasons for wishing to use a bidirectional grammar, some of which are listed below.

(a) The use of shared linguistic knowledge avoids the inefficiency of having distinct encodings of the same information.

(b) A system which uses a bidirectional grammar for grammar-based interpretation and tactical generation will be able to generate any sentence it can interpret and vice versa, thus avoiding inconsistencies in linguistic coverage between input and output.

(c) If a generation grammar which contains functional feature information is used for parsing, then the thematic organisation of the sentence is computed automatically during the parsing process. If a separate interpretation grammar is used which does not contain this information, then some additional code is required to extract the functional features from the parse tree and the logical form of the input sentence.

The use of a bidirectional grammar in a question answering system imposes a number of constraints on the design of the system. The most important of these is that the grammar must be entirely declarative (Appelt, 1987). This is achieved in our system by using a unification grammar consisting of a set of phrase structure rules whose categories are feature complexes.

The other constraint which applies to the grammar itself is that it must define a one-to-one mapping between unambiguous sentences and representations of their literal meanings. This is achieved by encoding the functional features described in Section 2 into the rules of the grammar.

A further constraint concerns the generation of sentences containing anaphoric forms. The logical form returned by the literal interpretation process is the skeletal logical form

in which anaphora are still unresolved. Hence, to generate sentences containing anaphoric forms in a system which uses a bidirectional grammar, it is first necessary to derive the skeletal logical form of the sentence from the full logical form produced during the strategic generation process. This is achieved by treating the transition from full logical form to skeletal logical form as the last stage of strategic generation.

It is also necessary to consider what has become known as "The Problem of Logical Form Equivalence" (Appelt, 1987; Shieber, 1988; Calder, Reape and Zeevat, 1989). This occurs when the tactical generator is presented with a logical form which is logically equivalent to but syntactically distinct from one for which the grammar defines a set of surface forms. For example, this may occur because logical connectives are associative and commutative, as in the examples shown below. The same problem occurs with predicates and functions which impose no ordering on their arguments.

(a) $\exists x,e.give(john,mary,x,e) \wedge book(x) \wedge past(e)$
(b) $\exists x,e.book(x) \wedge give(john,mary,x,e) \wedge past(e)$
(c) $\exists x,e.book(x) \wedge past(e) \wedge give(john,mary,x,e)$

The problem of logical equivalence also occurs with stative sentences like "Trinity is a college" if the stative verb is translated as the predicate 'equal,' as shown below. The form defined by the grammar would be (a), but the form chosen by an utterance planner with no access to the grammar would probably be (b).

(a) $\exists x.college(x) \wedge equal(x,trinity)$
(b) college(trinity)

However, as Shieber (1988) points out, the notion of equivalence required for natural language generation is not normal logical equivalence, but a finer-grained notion of "intentional equivalence." For example, the two logical forms shown below are logically equivalent, but they are not necessarily intentionally equivalent.

(a) college(trinity)
(b) $college(trinity) \wedge [dog(fido) \vee \neg dog(fido)]$

In general, the notion of intentional equivalence is quite difficult to define. For example, the two logical forms shown below are certainly logically equivalent, but it is unclear whether or not they are intentionally equivalent for our purposes.

(a) $dog(fido) \rightarrow animal(fido)$
(b) $animal(fido) \vee \neg dog(fido)$

On the one hand, if a speaker intends the content of (a) then he also intends the content of (b); but on the other hand, if (a) and (b) are presented to a tactical generator which uses standard unification, it is almost certain that different lexical items will be chosen to express the content.

In the current system the problem of logical form equivalence is effectively avoided by arranging the semantics of the grammatical rules such that the logical forms produced by the strategic generator have a standard logical syntax; work towards a better solution is described in Section 8.

4. The Grammar

The grammar used in the system is a phrase structure grammar augmented with feature unification, such as might be readily implemented in PATR-II (Shieber, 1986). The semantics of the grammar are specified using a simplified version of Montague semantics (Montague, 1974; Dowty, Wall and Peters, 1981). The desirability of this approach to semantics has been noted by a number of researchers (for example, see Hirst, 1983) and a number of systems have been successfully implemented which use this approach for interpretation (for a recent example, see Crabtree et al., 1988). The basic idea can be described as follows. To each rule of syntactic composition there corresponds a structurally analogous rule of semantic composition expressed in a typed higher-order logic. An example rule is shown below (feature values prefixed with '?' are variables).

Syntax: S [tense ?t] → NP [count ?x] VP [tense ?t, count ?x]
Semantics: $\exists e.\{NP\}(\{VP\}(e))$

The syntactic part of the rule says that a sentence may consist of a noun phrase followed

by a verb phrase, with count agreement between the two. Furthermore, the tense of the sentence is the same as the tense of the verb phrase. The semantic part of the rule says that the meaning of the sentence is an expression involving the meaning of the noun phrase and the meaning of the verb phrase. Some further examples are shown below.

Syntax: NP [count ?x] → Name [count ?x]
Semantics: λp.p({Name})
Syntax: VP [tense ?t, count ?x] → Vtr [tense ?t, count ?x] NP
Semantics: λe,x.{NP}(λy.{Vtr}(x,y,e))

Using a grammar composed of rules like these, sentences can be translated into their logical forms by parsing, applying the semantic rules compositionally, and finally applying lambda-reduction. Using expressions of higher-order logic in the semantics of the grammar poses special problems in the context of tactical generation; this will be discussed further in Section 6.

In order to use such a grammar for generation, some method of control over the choice of specific syntactic structures is required. There are various ways in which this can be achieved. The first approach, as used by Appelt (1985), is to invoke a decision-making routine each time a choice needs to be made. However, given this procedural element, it is difficult to see how such a grammar could also be used for parsing.

The second approach, used by Danlos (1987), is to control surface form by choosing specific syntactic forms before constructing the sentence. The choice of form is determined by the verb of the sentence and the rhetorical relation to be expressed. Surface forms are built up around "simple sentences" which are chosen from a list of possibilities called a "discourse grammar."

The final approach, suggested by McKeown (1983), is to encode the functional features into the grammar itself. This way, the data is kept declarative, and no additional control mechanism is required. This is the approach that we have adopted. The grammar presented here expands on McKeown's work by using a better developed syntactic analysis and a wider range of functional features.

The values of the functional features must allow them to identify parts of the logical form. For example, the feature "focus" may need to refer to the "mary" of the logical form ∃e.loves(john,mary,e). For this, the values "agent," "patient," "goal," "predicate" and "modifier" are used. The first three are common case names, and refer to the appropriate argument of the main proposition. The "predicate" value refers to the predicate of the main proposition, and "modifier" to a modifying clause. This scheme is clearly a simplification, but it is sufficient for the current grammar. It cannot cope with more than one modifier, for instance, and the functional features have no control over the thematic form of relative clauses. The use of these values is demonstrated below.

(a) It was founded by Henry-VI. [theme = patient, focus = agent]
(b) Henry-VI founded King's in 1441. [theme = agent, focus = modifier]
(c) The man who founded Kings built a library. [theme = agent, focus = patient]
(d) King's, Henry-VI founded. [theme = patient, focus = predicate]

The treatment of the tense feature has already been described; it is instantiated by the tense of the verb phrase. The treatment of the sentence type and emphasis features is straightforward, since these can be encoded as features of the sentence nodes of the appropriate rules. The treatment of theme and focus is slightly more complex. As a starting point, the following top level sentence rule can be used:

S [theme ?t, focus ?f] → NP VP [theme ?t, focus ?f]

This allows the verb phrase rules to decide the feature values. For an active VP without an attached PP, the theme is the agent and the linguistic focus is the patient. The values are reversed if the sentence is passivised. If the VP consists of an active VP with an attached PP, the theme is the agent and the focus is the modifier.

However, the situation is made more complex by the possible presence of gaps in the verb phrase. This requires the use of additional features, namely "potential focus" and "alternative focus." In the case of an active verb phrase, the potential focus is the patient, with the alternative focus being the predicate. The focus is instantiated as the potential focus if no gap is introduced and as the alternative focus otherwise. Hence, the

rule for an active verb phrase is as follows:

VP [theme agent, focus ?f] → Vtr NP [focus ?f, p-focus patient, a-focus predicate]

The value of the focus now depends on whether there is a gap in the noun phrase or not. The noun phrase will normally be a referring expression for the potential focus. This value is part of the data describing the gap, and is included the gap description by the use of the feature "gap-id." The rule which creates the gap is shown below.

NP [focus ?f, p-focus ?p, a-focus ?f, gap [syncat NP, gap-id ?p]] → ϕ

When the gap is consumed, for example, by the topicalisation rule given below, this identity becomes the value of the theme (see sentence (d) above).

S [theme ?t, focus ?f] → NP S [focus ?f, gap [syncat NP, gap-id ?t]]

By annotating the grammar rules with these functional features, the behaviour described in Section 2 can be produced.

5. Strategic Generation

The strategic generation component of the system consists of three parts: a theorem proving module called Deduce (originally implemented by Steve Pulman) with its associated database of domain knowledge; a set of question answering heuristics which construct the full logical form of the sentence to be generated plus an augmented set of functional features; and a module which converts the full logical form and augmented functional feature set into the logical message of the output sentence.

Deduce is a forward and backward chaining inference system, similar to that described by Charniak and McDermott (1985). Incoming expressions of first-order logic are converted to Kowalski normal form (Bundy, 1983) and then added to or tested against an associative database containing atomic propositions and inference rules. If the input is a query then a list of bindings for the variables contained in the query is returned.

5.1 Treatment of Wh-Questions

In building the question answering module, our aim was to investigate to what extent the information contained in the functional features of the user's question can be used to generate a cooperative response. In general, this is only one of the factors which needs to be taken into account in response generation – Kaplan (1983) and Webber (1987) discuss the variety of factors which can affect this process.

The treatment of wh-questions is fairly straightforward – a single call is made to the theorem prover and the output logical form is constructed on the basis of the binding list returned. The functional features are instantiated so that the linguistic focus of the response is the new information and the theme of the response is the same as the theme of the question (if possible). The question answering module also constructs a list of the entities which are present in both the original question and the response. This information is treated as an additional functional feature and is used to assist in the replacement of given entities by appropriate pronouns. An example of this process is given below.

Q: Who was King's founded by? [theme = kings]
Logical form of question = $\exists x,e.person(x) \wedge found(x,kings,e)$
Binding for x returned by Deduce = henry-vi
Full logical form of response = $\exists e.found(henry\text{-}vi,kings,e)$
A: It was founded by Henry-VI. [theme = it, focus = henry-vi]

If Deduce returns more than one entity as the answer to the question, logical forms are created for all the individual answers and these are conjoined to form the full logical form of the response. At present, the question answering module cannot deal with wh-questions for which no answers are found in the database, as it seems that this type of question presupposes that the information requested is held in the database. A simple solution to this problem is to assume negation by failure, which is adequate for the following example:

Q: Which colleges have a swimming pool?
A: No college has a swimming pool.

However, in some cases this will result in some rather strange answers being generated:

Q: Who designed King's?
A: Nobody designed King's.

This problem could perhaps be solved by asserting that "Every college was designed by somebody," which would result in a slightly better answer:

Q: Who designed King's?
A: I don't know who designed King's.

However, it seems that this problem needs to be investigated more fully before any definite conclusions can be drawn.

5.2 Treatment of Yes/No Questions

The algorithm for constructing logical forms for yes/no questions first makes an initial call to the theorem prover to see if the propositional content of the question is true. If it is, a simple affirmative answer is constructed, with the theme of the response being the same as the theme of the question. VP-ellipsis is applied during the transition from full logical form to skeletal logical form. An example of this is shown below.

Q: Was King's founded by Henry-VI? [theme = kings]
Logical form of question = ∃e.found(henry-vi,kings,e)
Answer returned by Deduce = yes
Full logical form of response = Yes, ∃e.found(henry-vi,kings,e)
A: Yes, it was. [theme = it]

If the initial yes/no question fails, it is turned into a wh-question by replacing the focus of the question with a wh-entity. The reason for this was postulated in Section 2 – the focus of the question signals the information that the user is unsure about. However, this alone is not sufficient, as it can result in some inappropriate answers if applied in this simple-minded fashion:

Q: Does King's have a library?
A: No, it has a chapel.

What is required here is a notion of semantic similarity, as demonstrated by the appropriateness of the following example:

Q: Does King's have a church?
A: No, it has a chapel.

An adequate treatment of semantic similarity is possible for atomic entities (i.e. proper nouns like "Wren"), but the cases shown above are more problematic. In its current state, the question answering module provides a simple negative answer if the linguistic focus of the input question is not an atomic entity. If the linguistic focus is atomic, it is assumed that the "slot" that the linguistic focus fills can only be filled by a particular kind of entity – for example, the first argument to the predicate 'found' will be an entity that is capable of founding things and the second argument will be something that is capable of being founded. This treatment is adequate for predicates that have a single well-defined meaning, as in the example below.

Q: Was King's founded by Wren? [focus = wren]
Logical form of question = ∃e.found(wren,kings,e)
Answer returned by Deduce = no
Logical form of wh-question = ∃x,e.found(x,kings,e)
Binding for x returned by Deduce = henry-vi
Full logical form of response = No, ∃e.found(henry-vi,kings,e)
A: No, it was founded by Henry-VI. [theme = it, focus = henry-vi]

If the wh-question constructed by replacing the linguistic focus with a wh-entity fails, then further wh-questions are constructed by replacing each of the remaining entities in the logical form of the input question with a wh-entity, and testing each of these questions against the database. If one of these questions succeeds, then it is marked as being the linguistic focus of the answer, with emphasis. An example of this is given below.

Q: Did Wren found King's? [theme = wren, focus = kings]
Logical form of question = ∃e.found(wren,kings,e)
Answer returned by Deduce = no
Logical form of wh-question = ∃x,e.found(wren,x,e)
Binding for x returned by Deduce = nil
Logical form of wh-question = ∃x,e.found(x,kings,e)
Binding for x returned by Deduce = henry-vi
Full logical form of response = No, ∃e.found(henry-vi,kings,e)
Linguistic focus of response = henry-vi, with emphasis
A: No, it was Henry-VI who founded King's.

If all the constructed wh-questions fail, we return to the problem reported earlier, where a wh-question returns an empty list of bindings. Again, negation by failure is a possible solution, and this is adopted in the current version of the system.

5.3 Computing Skeletal Logical Form

The full logical form and augmented feature set returned by the question answering module is mapped onto the logical message to be passed to the tactical generator using a fairly simple algorithm. In general, a significantly more complex algorithm than that presented here would be required. In the case of unmarked sentences, all given entities are replaced by pronouns of appropriate gender, as in the example given below.

Q: Did Henry-VI found King's in 1578?
A: No, he founded it in 1441.

If the emphasis feature is set, then no entities are replaced by pronouns. This means that the question "Did Wren found King's?" is answered by (a) rather than the slightly less natural (b):

(a) No, it was Henry-VI who founded King's.
(b) No, it was Henry-VI who founded it.

VP-ellipsis is dealt with by constructing a logical form containing a predicate which realises as the appropriate verb phrase anaphor. This is done if the linguistic focus of the response is nil. The original verbal predicate, together with the theme and tense of the

response, are used to determine which predicate should be used.

6. The Tactical Generator

The tactical generation component of the system consists of two modules. The first maps the logical message returned by the strategic generator onto the logical message required by the grammar. The second module is the sentence generator, which operates by forming trees top-down, breadth-first. Each node of a tree represents a syntactic constituent. This process is guided by the values of the functional features, but some semantic information from the goal logical form is also used to cut down the search space.

The first of these modules operates by using information contained in the grammar to find the main clause of the logical form. This is done by finding the clause which can realise as a verb and which contains the theme and the linguistic focus. The predicate, arguments and modifier of this clause are determined and the theme and linguistic focus of the logical message are updated accordingly. The reverse mapping is performed during interpretation. An additional set of features can be computed from the logical form at this stage to cut down the search space required by the sentence generator. For example, if the agent of the main clause is nil, the feature 'agent' can be set to nil, which results in an agentless passive being generated.

The process of sentence generation in our system is a complex task, due to our use of Montague semantics in the grammar. This means that a bottom-up algorithm using standard unification to compare logical forms, such as that used by Shieber et al. (1989) is not feasible. For the current system, we have therefore adopted a standard top-down, feature-driven algorithm.

The sentence generator takes as input a logical form and a set of sentence level feature instantiations, from which an initial tree is created. The sentence level node of this tree is then selected for expansion. This proceeds by finding all rules whose mothers unify with the node, subject to the restriction that the semantics of the rule and the goal logical form are not incompatible. For example, if the goal logical form is

$\exists$e.love(john,mary,e) and we are extending a node of category Vtr [count sing] then only rule (a) would be selected.

(a) Syntax: Vtr [count sing] $\rightarrow$ loves
    Semantics: love
(b) Syntax: Vtr [count plur] $\rightarrow$ love
    Semantics: love
(c) Syntax: Vtr [count sing] $\rightarrow$ likes
    Semantics: like

The tree is then extended with the first rule that applies, with further trees being added to the agenda if more than one rule is found to be applicable. As the tree is extended, the associated logical form is constructed using the semantic part of the rule being used. For example, if the goal logical form is $\exists$e.found(henry-vi,kings,e), the top level node is first expanded using rule (a) below. At this point the semantics is $\exists$e.{node1}({node2}(e)), with node1 representing the noun phrase and node2 the verb phrase. The verb phrase is then extended using rule (b), after which the semantics of the tree is $\exists$e.{node1}(λe,x.{node4}(λy.{node3}(x,y,e))(e)), with node3 representing the transitive verb and node4 the second noun phrase.

(a) Syntax: S $\rightarrow$ NP VP
    Semantics: $\exists$e.{NP}({VP}(e))
(b) Syntax: VP $\rightarrow$ Vtr NP
    Semantics: λe,x.{NP}(λy.{Vtr}(x,y,e))

This process continues until the tree has extended down to the lexical level, at which point lambda reduction is applied to the constructed logical form. This reduced logical form is then compared with the goal logical form using a unification algorithm which only allows variables to match against other variables. This implies that, if the unification succeeds, the goal logical form always subsumes the constructed logical form and vice versa. This ensures that the generation algorithm is both coherent and complete, as defined by Wedekind (1988).

7. A Worked Example

This section aims to show how the system works as a whole by presenting a simple worked example. The question and answer pair to be used for this example is shown below.

Q: Was King's founded in 1578?
A: No, it was founded in 1441.

7.1 The Interpretation Phase

The input question is parsed and translated into its logical form using the grammar-based interpretation facilities provided by the Grammar Development Environment. The features attached to the top level node of the parse tree are extracted, and these are used, together with the logical form, to set up the logical message of the input. The output from this phase is shown below.

Logical form = $\exists x,e.found(x,kings,e) \wedge in(e,1578)$
Sentence type = yes/no question
Tense = past
Theme = kings
Focus = 1578
Emphasis = nil

7.2 The Strategic Generation Phase

The first step in the strategic generation phase is to extract all possible information from the input and use this to construct an initial version of the output message. All the entities are extracted from the logical form of the question and this is stored as an additional field in the message. The theme and tense are copied from the input to the output, the sentence type is set to declarative, and the logical form, focus and emphasis fields are set to nil. The resulting message for our example is given below.

Logical form = nil
Given entities = {kings,1578}
Sentence type = declarative
Tense = past
Theme = kings
Focus = nil
Emphasis = nil

The main question answering routine is then entered. For our example, the initial yes/no question fails, but the wh-question constructed by replacing the focus with a wh-entity succeeds, and returns the answer 1441. The logical form of the response is then constructed, 1578 is removed from the list of given entities and 1441 becomes the focus, as shown below.

Logical form = No, $\exists$x,e.found(x,kings,e) $\wedge$ in(e,1441)
Given entities = {kings}
Sentence type = declarative
Tense = past
Theme = kings
Focus = 1441
Emphasis = nil

The last step of the strategic generation phase maps the full logical form onto the skeletal logical form, as described in Section 6. In the case of our example, this means that the final logical message returned by the strategic generator is as follows:

Logical form = No, $\exists$x,e.found(x,it,e) $\wedge$ in(e,1441)
Sentence type = declarative
Tense = past
Theme = it
Focus = 1441
Emphasis = nil

7.3 The Tactical Generation Phase

The first part of the tactical generation phase maps the feature set shown above onto the feature set required by the grammar. This means that the theme now has the value

"patient" and the linguistic focus is "modifier." The auxiliary feature set is computed from the logical form, and this is added to the functional feature set to form the starting category of the sentence. Sentence generation then begins, as described in the previous section. For each tree that proceeds down to the lexical level, the constructed semantics is lambda reduced and compared with the goal logical form. For our example, the sentence "No, it was founded in 1441." is produced. Due to the restrictions placed on the generation process by the feature specifications, only eight trees are produced, and the sentence generation process takes just under 0.7 seconds of CPU time to run to completion.

8. Further Research

The most obvious deficiency of the current system is that it only deals with question/answer pairs, and that the cooperativity of the system is limited to the use of the thematic organisation of the user's input. While this needs to be taken into account, it is certainly not the only factor that needs to be observed in the construction of a cooperative response. For example, if a hungry user approaches the system and asks "Is the Pizza Hut open today?" a response of "No, it's open tomorrow" is inappropriate. A better response, based on the expected plan of the user, would be "No, but Pizzaland is." Work is currently under way to integrate the theory of question answering reported here with plan-based models of language use, such as those reported by Allen (1987) and Appelt (1985). We envisage that this can be done bidirectionally, with plan recognition and utterance planning sharing the same sources of knowledge about actions, the domain, and the beliefs and plans of the user. This will also allow the system to deal with dialogues consisting of more than a single question/answer pair.

The problem of logical form equivalence is avoided in the current system by arranging that the responses returned by the strategic generator have the correct logical syntax. However, in general this cannot be ensured, and so a better solution is urgently required. We are currently developing an algorithm for comparing two logical forms which involves computing a sorted clausal form of the two logical expressions before

unification is applied. When it is complete, this algorithm will be used in the top-down version of the generator in place of the unification routine reported here.

The current version of the generator places a heavy burden on the functional features for efficient operation, and is limited in the class of grammars for which it is applicable, as discussed by Shieber et al. (1989). Work is under way on the design and implementation of a bottom-up generator for use with the formalism described in this paper. The proposed matching routine involves compiling the semantics of the goal logical form into a set of specifications, to allow for certain forms of logical equivalence. The proposed bottom-up algorithm also takes more account of lexical semantics.

Semantic, syntactic and lexical considerations can block the use of certain thematic forms. For instance, some verbs can only be realised in the active, as in the examples below.

(a) John married Mary.
(b) John perjured himself.
(c) King's has a chapel.

Also, passivisation of transitive verbs and indirect object transformation with ditransitive verbs should be blocked for generation in cases likes the following:

(a) ? A chicken was eaten by John.
(b) * John took the zoo his son.

A possible explanation for phenomena such as these is given by Siewierska (1984), in terms of the animacy and definiteness of the entities involved in the sentence. Work is currently being undertaken to encode restrictions such as these into the grammar so that the sentences produced by the generator are both natural and appropriate to their discourse context.

Acknowledgements

We would like to thank our supervisor, Steve Pulman, for his expert advice and for his excellent lecture notes on syntax and semantics, from which many of the basic treatments

used in the system are taken. We also extend our thanks to Derek Bridge, John Carroll, Marianne McCormick and David Milward for many useful discussions, and to Karen Sparck Jones, David Milward, Steve Pulman and Alethea Tabor for their comments on earlier drafts of this paper. The authors are supported by studentships from the Science and Engineering Research Council.

References

Allen, J. F. (1987) *Natural Language Understanding*, Benjamin/Cummings.

Appelt, D. E. (1985) *Planning English Sentences*, Cambridge University Press.

Appelt, D. E. (1987) "Bidirectional Grammars and the Design of Natural Language Generation Systems," *Position Papers for TINLAP-3*, Association for Computational Linguistics, 206-212.

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

Briscoe, T., Grover, C., Boguraev, B. and Carroll, J. (1987) "A Formalism and Environment for the Development of a Large Grammar of English," *IJCAI-87*, 703-708.

Brown, G. and Yule, G. (1983) *Discourse Analysis*, Cambridge University Press.

Bundy, A. (1983) *The Computer Modelling of Mathematical Reasoning*, Academic Press.

Calder, J., Reape, M. and Zeevat, H. (1989) "An Algorithm for Generation in Unification Categorial Grammar," *Proceedings of the Fourth Conference of the European Chapter of the ACL*, 233-240.

Carroll, J. Boguraev, B., Grover, C. and Briscoe, T. (1988) *A Development Environment for Large Natural Language Grammars*, University of Cambridge, Computer Laboratory, Technical Report No. 127.

Charniak, E. and McDermott, D. V. (1985) *Introduction to Artificial Intelligence*, Addison-Wesley.

Crabtree, B., Crouch, R. S., Moffat, D. C., Pirie, N., Pulman, S. G., Ritchie, G. D. and Tate, A. (1988) *A Natural Language Interface to an Intelligent Planning System*, DAI Research Paper No. 407, Department of Artificial Intelligence, University of Edinburgh.

Danlos, L. (1987) *The Linguistic Basis of Text Generation*, Cambridge University Press.

Dowty, D. R., Wall, R. and Peters, S. (1981) *An Introduction to Montague Semantics*, D. Reidel.

Gazdar, G., Klein, E., Pullum, G. K. and Sag, I. A. (1985) *Generalized Phrase Structure Grammar*, Basil Blackwell.

Hajicova, E. (1983) "Topic and Focus," *Theoretical Linguistics 10*, 268-276.

Hajicova, E. (1987) "Focussing – A Meeting Point of Linguistics and Artificial Intelligence," in Jorrand, P. and Sgurev, V. (eds.) *Artificial Intelligence II: Methodology, Systems, Applications*, Elsevier Science Publishers, 311-321.

Hajicova, E. and Sgall, P. (1985) "Towards an Automatic Identification of Topic and Focus," *Proceedings of the Second Conference of the European Chapter of the ACL*, 263-267.

Halliday, M. A. K. (1967) "Notes on Transitivity and Theme in English, Part 2," *Journal of Linguistics 3*, 199-224.

Hirst, G. (1983) "A Foundation for Semantic Interpretation," *Proceedings of the 21st Annual Meeting of the ACL*, 64-73.

Jacobs, P. S. (1988) "Achieving Bidirectionality," *Proceedings of the 12th International Conference on Computational Linguistics*, 267-269.

Kaplan, S. J. (1983) "Cooperative Responses from a Portable Natural Language Database Query System," in Brady, M. and Berwick, R. C. (eds.) *Computational Models of Discourse*, MIT Press, 167-208.

Levinson, S. C. (1983) *Pragmatics*, Cambridge University Press.

McKeown, K. R. (1983) "Focus Constraints on Language Generation," *IJCAI-83*, 582-587.

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

Montague, R. (1974) "The Proper Treatment of Quantification in Ordinary English," in Thomason, R. H. (ed.) *Formal Philosophy: Selected Papers of Richard Montague*, Yale University Press, 247-270.

Quirk, R., Greenbaum, S., Leech, G. and Svartvik, J. (1985) *A Comprehensive Grammar of the English Language*, Longman.

Rosenschein, S. J. and Shieber, S. M. (1982) "Translating English into Logical Form," *Proceedings of the 20th Annual Meeting of the ACL*, 1-8.

Shieber, S. M. (1986) *An Introduction to Unification-Based Approaches to Grammar*, Center for the Study of Language and Information.

Shieber, S. M. (1988) "A Uniform Architecture for Parsing and Generation," *Proceedings of the 12th International Conference on Computational Linguistics*, 614-619.

Shieber, S. M., van Noord, G., Moore, R. C. and Pereira, F. C. N. (1989) "A Semantic-Head-Driven Generation Algorithm for Unification-Based Formalisms," *Proceedings of the 27th Annual Meeting of the ACL*.

Siewierska, A. (1984) *The Passive: A Comparative Linguistic Analysis*, Croom-Helm.

Webber, B. L. (1987) "Question Answering," in Shapiro, S. C. (ed.) *Encyclopedia of Artificial Intelligence*, John Wiley, 814-822.

Wedekind, J. (1988) "Generation as Structure Driven Derivation," *Proceedings of the 12th International Conference on Computational Linguistics*, 732-737.