Labelled Abduction and Relevance Reasoning

Dov Gabbay
Jeremy Pitt[*]

Department of Computing,
Imperial College,
London, SW7 2BZ
email {dg,jvp}@doc.ic.ac.uk

Ruth Kempson

Department of Linguistics,
School of Oriental and African Studies,
London, WC1H 0XG
email kempson@clus1.ulcc.ac.uk

Abstract

The aim of this paper is to explore modelling a more non-standard interaction between a human and a database. We try to equip the database with the capability of understanding queries in natural language and answering them in a more 'intelligent' manner than by mere yes/no factual answers. This capability of the database requires (at least) two components. First the database must possess a front-end logic module which can parse the natural language input I and translate it into a query (or update) Q_I in its own (logical) query language. Second, and in fact more difficult, it must cope with the logical patterns of human answer/input/query, which is logically more (non-monotonic) common sense than ordinary query languages.

In this paper, we first address the preliminary parsing/translation component. We present an approach to utterance interpretation which takes as input sequences of natural language words and builds linked labelled databases as output, and also involves additional abductive choices which are made on-line and captured through the labelling algebra. We describe the outlines of the formal model we are developing, and provide an experimental implementation of the on-line construction of linked labelled databases and interpretation relative to a context. We then turn to discussion of the second component, and consider in detail the formal properties of the abduction principles required to equip a database with the 'common sense' faculties required to participate in question-answering and dialogues.

1 Introduction

The aim of this paper is to explore modelling a more non-standard interaction between a human and a database. We try to equip the database with the capability of understanding queries in natural language and answering them in a more 'intelligent' manner than by mere yes/no factual answers. This capability of the database requires two components. First the database must possess a front-end logic module which can parse the natural language input I and translate it into a query (or update) Q_I in its own (logical) query language. Second, and in fact more difficult, it must cope with the logical patterns of human answer/input/query, which is logically more (non-monotonic) common sense than ordinary query languages.

To exemplify informally the problems we seek to address, we will first consider what needs to be added to the process of answering a database query in a way that reflects human reasoning in response to natural language input. Take, to begin with, the following extract from a simple Prolog-like database Δ_1, which, let us assume, can be accessed via some machine much like a cash dispenser:

[*]Supported by the Economic and Social Science Research Council under grant ESRC R-000-23-2069, and CEC Esprit BRA Medlar II (Esprit 6471).

$$\Delta_1 : \boxed{\begin{array}{l} Holiday(x,y) \text{ if } Available(y) \wedge Afford(x,y) \\ Afford(x,y) \text{ if } Cost(y,z) \wedge Credit(x,z) \\ Credit(x,z) \text{ if } In\_limit(x,r) \\ Sunnyplace(a_i), i = 1 \ldots 3 \\ Cost(a_1, 200) \\ Cost(a_2, 250) \\ Cost(a_3, 700) \\ Sunnyplace(b) \\ Cost(b, 500) \end{array}}$$

A customer accesses the database by inserting his visa card into the slot and typing the query (*holiday*, a_1, a_2, a_3). The database can automatically get from the visa card his name x_0, and other credit details, and asks itself the queries:

$$?Holiday(x_0, a_i), i = 1, 2, 3,$$

An ordinary non-intelligent database will simply check if each query succeeds and output the details if it does and output fail if not. A more intelligent database might output an explanation in case of failure, such as:

You can't have a_1 *because it is not available*

or:

You can't have a_3 *because you cannot get credit*

An even more intelligent database may recognise, for example, that none of the a_i are available but they are all sunny places and b is a sunny place within the nearest price range. So using a metaprogram of 'interest', 'view' etc, might ask the user (in whatever way it 'communicates'):

Would you like (holiday,b)

and output the details. We are sure many papers in this volume address many of the issues related to the above capabilities.

Ultimately, though, we would like the human to ask in natural language rather than type the 'agreed' form of input (e.g. (holiday, a)) and furthermore reply naturally to a computer query. This might transcend simple "yes" and "no" answers: for example, a 'natural' form of response to the computer-generated query above might be:

"b is expensive."

The computer should understand the English and deduce (as a human would) that the answer to the query is "no", *and* that any holiday costing more than \$500 is not a candidate. Ideally, we would like to imbue the intelligent database with the same faculty, so that it could now reply:

There are no cheaper holidays

to which a possible form of response might be:

OK, I will take it.

which includes anaphora, *it* referring to b, and withdrawal of the previous denial of b as a suitable holiday.

There are a number of hidden problems here, but essentially we need to spell out two processes: one the reasoning underlying the parsing process, including the interpretation of anaphoric expressions, the access of related information, etc.; and the other the reasoning underlying question-answer dialogues, which may involve indirect answers, negotiation requiring changes in beliefs, plans and goals, and so on.

In the first process, the process of *utterance interpretation*, there are two reasoning problems. Firstly there is the reasoning the hearer engages in to establish the form and content which the utterance directly expresses, and secondly there is the reasoning required for the construal of pronouns, adverbs of tense, etc. How these are construed depends on what is already established in the discourse, in some sense that needs to be explained. Moreover, as the array of examples (1)-(5) show, this phenomenon ranges from deciding on the value of a pronominal, through deciding on how to fix the proposition expressed in a flow of time, to reconstructing an entire structure on the basis of some provided fragment:

(1) *John came in. He had failed.*
(2) *Every student worried he had failed.*
(3) *If he had failed, John would be at home by now.*
(4) *John had failed. Bill too.*
(5) *If he had failed, John would be at home by now. Bill too.*

The second process, how a speaker intends, and can rely on, the hearer to retrieve considerably more from some utterance that merely what its words can be taken to express in the context, is clearly displayed in the use of an indirect answer. For example, consider the responses below of two dinner party guests to their host's question on arrival:

(6)	HOST	*Do you like Spaghetti Bolognese?*
(7)	GUEST 1	*I'm Italian.*
(8)	GUEST 2	*I'm vegetarian.*

The result of the two guests hearing (6) above leads them to deduce that dinner is Spaghetti Bolognese, although this is not explicitly stated. For the host hearing the indirect reply (7), he (or she) should retrieve the 'common knowledge' that all Italians like pasta, and therefore deduce the answer "yes", but on hearing (8) he should retrieve the 'common knowledge' that all vegetarians do *not* eat meat, and hence deduce the answer "no".

There is a general pattern to this variety of effects in natural language processing. Given some string of words as input, hearers systematically add to the information these words provide in an active process of structure-building and inference-drawing. If, then, we wish to model dialogue successfully, we have to reconstruct the reasoning that underpins this enrichment process in a maximally general way, applying it both to the decisions involved in establishing what a sentence expresses, and what additional information to draw from it.

This is the task of this paper. We adopt the general Labelled Deductive Systems LDS framework of [Gabbay, 1990], and use it to model the dynamic process of utterance interpretation whereby some database is created, and how the resulting database is then used in the more general process of on-line reasoning. In section 2 we describe the formal framework underpinning our model of utterance interpretation, and in section 3 we describe a prototype implementation of the parsing process in a restricted context. Section 4 considers more general reasoning processes, in particular abductive reasoning, involved in question-answering and dialogue processing in general. The parsing system is intended to be embedded within such reasoning processes, to provide a database with an intelligent capability allowing it to participate in question-answering and dialogues.

2 The Formal Framework

In this section, we briefly introduce the framework of Labelled Deductive Systems, and abductive mechanisms within this framework (sections 2.1 and 2.2 respectively). In section 2.3 we consider the general problem of parsing requiring two types of abductive reasoning for interpretation and inferential effect, and in section 2.4 we combine LDS with these ideas on parsing to develop a formal model of utterance interpretation.

2.1 Labelled Deductive Systems

Labelled Deductive Systems provide a uniform algorithmic characterization of a wide variety of logics. An LDS is a triple $<\mathbf{L}, \Gamma, \mathcal{M}>$, where:

i $\mathbf{L}$ is a logical language (connectives and well-formed formulas),

ii Γ is an algebra of labels, with some operations over the labels,

iii $\mathcal{M}$ is a discipline of labelling formulas of the logic $\mathbf{L}$ with elements from the algebra of labels Γ, together with deduction rules and agreed ways of propagating the labels *via* the application of the deduction rules.

The basic data item in LDS is the *labelled formula* $\alpha : P$, where $\alpha \in \Gamma$ is the label and $P \in \mathbf{L}$ is the formula. The role of the label is to provide additional information about the formula which is not of the same declarative nature of the formula itself; for example, α could represent the 'degree of reliability' of the formula P, or the 'time' at which the P was true, or the 'source' of the information represented by P. A collection of data items, i.e. a database, is no longer just a "set of assumptions", but can have a much more general structure, e.g. multisets, lists, partially-ordered sets, *etc.*

However, the algorithmic rules still define the notion of a consequence relation, only now between labelled formulas, i.e.:

$$\alpha_1 : P_1, \ldots, \alpha_n : P_n \vdash \beta : Q$$

We will be particularly interested in the computation of this consequence relation, which takes the general form:

$$\delta : \begin{array}{l} \textbf{prove } \{\alpha_1 : P_1, \ldots, \alpha_n : P_n\} \vdash \beta : Q \\ \boxed{\qquad\qquad\qquad \vdots \qquad\qquad\qquad} \\ \textbf{exit } \beta = \ldots \end{array}$$

i.e. we want to prove, from a given sequence of labelled formulas, a formula Q, and in doing so we want to compute the label of the formula (β) and the label of the proof (or database) itself (δ).

The way the inference rules are used in such computations (the box above represents the application of inference rules used in a computation) is more or less uniform to all systems. For example, for the language with implication, the rules are predetermined in their structure for all logics, with only some fixed and agreed room for variations. The $\rightarrow$–elimination rule (i.e. modus ponens) has the form:

$$\frac{\alpha : P \quad \beta : P \rightarrow Q}{\alpha + \beta : Q}\ [\rightarrow\text{-E}]$$

where $+$, a binary function symbol, is an operation over the algebra of labels. The $\rightarrow$–introduction rule has the form:

$$\begin{array}{l} \boxed{\begin{array}{ll} \alpha : P & \textit{assume} \\ \vdots & \\ \alpha + \beta : Q & \textit{show} \end{array}} \\ \textbf{exit } \beta : P \rightarrow Q \end{array}$$

which intuitively may be read as stating: to show $\beta : P \rightarrow Q$ start a subcomputation (again represented by the box), assume $\alpha : P$, and show $\gamma : Q$, with α, β and γ satisfying the equation $\gamma = \alpha + \beta$.

The methodology of LDS has provided a general framework for the formalisation of many aspects of practical reasoning [Cunningham *et al.*, 1991], including those where the reasoning involved in planning, co-operative problem solving *etc* is often manifest in natural language dialogues. However, it is not just the mechanisms that drive intelligent dialogues, but also natural language phenomena themselves (syntax, semantics, and pragmatics), that are amenable to analysis with LDS. In section 2.3 we discuss applying LDS to the parsing process itself.

2.2 Labelled Abductive Mechanisms

The basic *deductive* apparatus of LDS is a precise system of rules allowing one to show (or fail to show) whether $\Delta \vdash \Gamma$, for Δ a data structure and Γ a goal structure. Most useful among the goal structures is the unit structure of the form $t : G$. Thus for the purpose of explaining what abduction is going to be in our framework, we assume that the notion of:

$$\Delta \vdash t : G$$

is precisely (algorithmically) defined, thus yielding a particular LDS system $\mathbf{L}$. We now schematically explain how abduction principles fit into this framework. Consider a database Δ, containing $\alpha : X$ inside it, which we write as $\Delta[\alpha : X]$. Somewhere in the structure Δ, α is a label variable and X is a propositional variable standing for a wff. For any particular choice of X, say $X = A$ and $\alpha = \alpha_0$, $\Delta[\alpha_0 : A]$ is a proper database.

Suppose we want to prove a goal $t : G$. Then for some (maybe no) wffs A_i and labels α_i we may have:

$$\Delta[\alpha_i : A_i] \vdash t : G$$

A principle of abduction:

$$Abduce(\alpha : X)$$

is a computation (algorithm) that can choose one or more of the $\alpha_i : A_i$ above. Of course for different t and G we get different α_i and A_i.

The importance of the above point of view is:

1. Databases can take abductive principles as part of their data, slotted in at the right places;
2. The abductive principle is relative to the computation procedure and the rest of the database. Thus when new data is put in, the abductive principle changes.

A fuller account of abduction in LDS can be found in [Gabbay, 1991].

2.3 Utterance Interpretation: The Parsing Problem

We take as our background the informal insights of Relevance Theory [Sperber and Wilson, 1986] and their view of the psychological process of inference. Interpreting signals presented in a dialogue is taken to involve providing the essential preliminary for any such process to take place: i.e. by building a well-formed structure in some inference system. This structure is then used to derive further inferences using additional information chosen on the basis of the signal. The hearer's task is thus two-fold. She has to decode the signal with whatever her knowledge of words of her language provides (and as we have already seen, there is the additional problem that the intrinsic "content" of natural language signals characteristically underdetermines any such assigned "interpretation"); and she has to enrich that information by making additional choices. Both these tasks are carried out with a certain goal in mind, which is to derive extra inferential effect, i.e.:

1. establish a vehicle for inference Δ_k from the sequence of words;
2. combine Δ_k with some other information Δ_j to yield extra inferential effect Δ_m:

$$\Delta_k, \Delta_j \vdash \Delta_m$$

The *principle of relevance* at the heart of Relevance Theory is that these tasks are constrained to provide cognitively the best possible outcome - a set of inferential effects sufficiently rich to offset whatever cognitive effort is involved in making these additional choices Δ_j. Cognitive effort is always to be balanced by inferential benefit, which will, in general, mean using whatever information is most easily available to make these choices. Although we refer to this principle on occasion within this paper, its formalisation is not our primary concern. At this point we wish to set out a logical framework within which the study of such choices can be made.

Thus we reconstruct these tasks by natural deduction within the LDS framework. The overall goal is to achieve nontrivial inferences in some proof domain. The subgoal is to build a database capable of establishing the requisite well-formed inference structure to constitute the major premise in such a domain. These goals are achieved by taking the words in sequence, constructing a database from them, and using abduction to make additional choices whenever the information presented is insufficient. We wish to use this approach to model not only the extra information the hearer accesses to yield indirect implications but also the extra information the hearer accesses in resolving anaphora and temporal relations. The former involves abduction of whole premises of a relatively familiar sort. For the latter, we shall require a concept of *structural abduction*, for which we develop the idea that *abduction principles are data.*

For example, consider the following database and query:

a_1	A
a_2	$A \rightarrow (B \rightarrow S)$
a_3	B
...	...
a_4	X, abduce on structure to find X
a_5	$B \rightarrow D$

The goal is $?S + D$.

By writing $S + D$ for the goal we are saying we want to partition the database, which is a list of assumptions, into two parts, the first part must prove S and the second part must prove D. This is done in resource logics, where one pays attention to the question of which part of the database proves what.

Such considerations arise in many areas for example in what is known as *parsing as deduction* [Pereira and Warren, 1983]. As an example, consider the parsing problem presented by the natural language sequence: *Mary hit John. He cried.* The way this can be analysed is by assuming that each word is assigned a wff of some resource logic, corresponding to its type (i.e. the Curry-Howard interpretation of formulas as types), with a label which corresponds to the lexical item itself. We use here logical types defined inductively from basic types e (entities) and t (truth values) and the rule *if A and B are types then $A \rightarrow B$ is a type.*

Assignment is done at the lexical level. Thus a proper noun n is assigned $n : e$, an intransitive verb v_1 is assigned $v_1 : e \rightarrow t$, and a transitive verb v_2 is assigned $v_2 : e \rightarrow (e \rightarrow t)$. These assignments are commonly used in computational linguistics and in categorial grammar (see e.g. [Oehrle *et al.*, 1988]), and give us straightforward data – the building blocks of the database. The pronoun *he*, on the other hand, is assigned an abduction principle. Our problem then becomes:

Data
1. *Mary: e*
2. *hit: $e \rightarrow (e \rightarrow t)$*
3. *John: e*
4. *he:* Abduce on structure: Take the first literal up the list.
5. *cried: $e \rightarrow t$.*
Query
Prove $?S$ or $?S + S$ or $?S + S + S \ldots$ etc

We are thus saying that anaphora resolution makes use of structural abduction, and that a database can either display data items or give us pointers to where to get them. Thus a labelled database can look like:

$$\boxed{\begin{array}{ll} n_1\text{:} & \text{data } m \\ \vdots & \vdots \\ n_k\text{:} & \text{get datum from} \ldots \\ \vdots & \vdots \end{array}}$$

From the logical point of view we are using the following principle:

- Abduction principles can serve as items of data in one database Δ_0, being a pointer to another database Δ_1 where some computation is carried out jointly involving both databases and the result is the abduced data item in Δ_0.

2.4 Language and Reasoning

With this simple and highly idealised example, we nonetheless have the basis of our approach to natural language processing, which involves building structured databases by deduction over declarative units made up of a word as label and a formula, making additional abductive choices as necessary. The result of this goal-driven parsing task is a predicate logic formula which can then perform a role in more general inference tasks. Furthermore, we want that all decisions to be made at each stage of database building or inference drawing are driven by the need to achieve the goal of deriving inferential effects relative to some constraint of 'minimal effort'.

The natural language input itself provides the trigger to this process of interpretation. Some words provide the premises themselves, some provide procedures which determine where further requisite information may be found, while others dictate the form of the goal to be achieved. The common property that words share is that they provide procedures for the hearer to follow, which, if followed, are intended to guarantee the right choice of well-formed output. That is, the hearer will recover the information the speaker intended her to - modelled in our approach by a predicate logic formula derived as the conclusion of some linked labelled database.

With these informal ideas to hand, we can now set out our model of the parsing process. The model we are developing (called L&R (Language and Reasoning): see also [Gabbay and Kempson, 1992]) merges aspects of grammar, parsing and interpretation, and consists of:

i *a lexicon,* in which each word of the language is associated with declarative and procedural information, more specifically a triple <*labels, formulas, instructions*>, where the labels and formulas are declarative information and the instructions are procedural.

$$< \underbrace{\textit{labels, formulas,}}_{\textit{declarative}} \overbrace{\textit{instructions}}^{\textit{procedural}} >$$

ii *a logic,* which is an LDS, and so defines *declarative units*, each of which is a pair consisting of a label and a formula; and *linked labelled databases*, where a database is an ordered collection of declarative units and can be linked by shared variables.

iii *an algorithm,* which given a linked database and a triple as input, constructs a new linked database by adding the declarative unit specified by the triple; the result is computed by the interaction of the instructions specified by the triple, the inference rules of the logic, and the algebra of the labels.

Parsing a sentence in L&R is a matter of proving that a well-formed linked database of the form $\alpha : t$ results from a given input sequence of words, which are lexically analysed into triples. This linked database also constitutes a representation of the meaning of that input. It is also used in 'abductive enrichment': the choice of additional databases as premises which combined yield inferential effect.

We exemplify the building of the initial database by considering the steps involved in interpreting the utterance *"John likes Mary, who he admires"*. A natural deduction proof of parsing/interpreting this utterance in the L&R model is shown in figure 1 (see page 9), and the following description is best read with reference to this figure.

The goal is to prove that a formula $\alpha : t$ follows from a given sequence of assumptions. The proof itself is a labelled database, the sequence of assumptions is the labels and formulas of the words in the utterance, which also provide instructions for manipulating these data items.

The words *John* and *Mary* each provide a pair comprising a label (the word itself) and a formula (e). These are simply inserted in the database s_1 at steps 1 and 4, and are used as minor premises in the chain of reasoning involved in the interpretation. The verbs *like* and *admire* constitute the major premises in their respective databases, and also carry instructions about putting side-conditions on the proof steps (i.e. 'use_me_last' and 'use_me_first' in steps 1, 4, 5 and 9, dictating that $John\colon e$ is used last in some chain of premises leading to $\alpha : t$, and conversely $Mary\colon e$ is used first) and the labelling of the database itself (steps 3 and 8). The side-conditions are used in the modelling of subject and object relations, and (here) the database label indicates that the temporal flow against which the database is constructed includes the time of evaluation.

Unlike standard analyses in terms of some higher type specification (e.g. [Morrill, 1990]) the word *who* is modelled directly as projecting the procedural instruction that a new database (here s_2) should be built but with a more specific goal, i.e. that the label of the formula t should be linked to the current database through a shared label (v). Finally, *he* provides a data item similar to *John* and *Mary*, although the label is a metavariable: the word also comes with the instructions on how to identify the expression that instantiates the metavariable in step 6.

The goal specification of s_2 and the lexical (type) specification of admire dictate that $v\colon e$ is assumed in step 9. The remaining steps 10, 11, 12 and 13 are then applications of modus ponens as in section 2.1, with the '+' operator on labels being function-argument application. The resultant linked labelled databases are as shown in figure 1. It then remains to complete the mapping from a sequence of English words onto some predicate logic formula, for details of which see [Gabbay and Kempson, 1992].

Many of the details presented in the proof configuration displayed in figure 1 require detailed linguistic justification *cf* [Gabbay and Kempson, 1992]. In particular, there is no account here of quantification. We simply presume noun phrases project a premise of the form $\alpha : e$, with *who* not projecting a premise into the database but a goal specification on its outcome. We are concerned merely to demonstrate how from a set of lexical specifications and assumptions of natural deduction set within an LDS framework, we can model the parsing process as a process which simultaneously provides a recognition device and an interpretation builder. Note in particular that the parsing process is not a trivial decoding of some signal. Essential steps of reasoning are involved every step of the way.

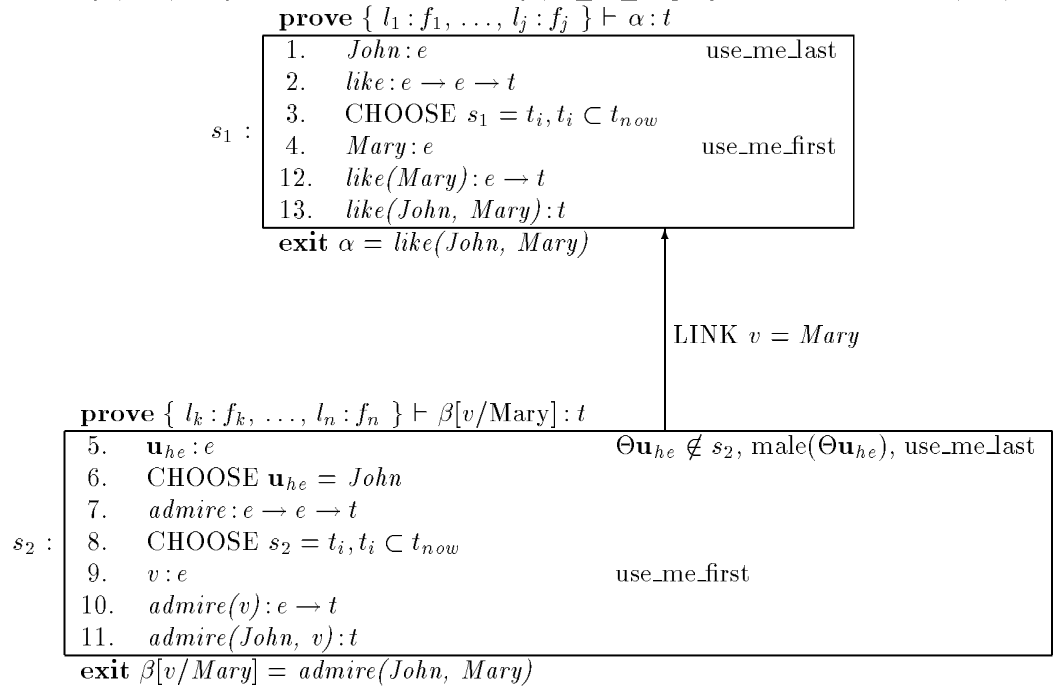
3 An Experimental Implementation

This section describes a prototype implementation of the formal model described in section 2.4. Section 3.1 discusses the implementation of a parser for the model, section 3.2 describes a restricted context (a blocks world), and section 3.3 describes how the parsing process can be embedded in such a context to provide an implementation of utterance interpretation relative to a context. We should stress that the significance of this particular application lies not in the implementation itself, but rather in the way in which deductive systems constructed for logical purposes can be applied equally in natural language parsing, natural language interpretation, and planning tasks.

3.1 L&R Parsing

Although the algorithm for constructing sentence-meanings is intended to be *procedural*, our experimental implementation has used our own externalised knowledge of grammar rules, although the *combinatorial* information specified in such rules is ignored completely. The implementation

For $l_1 : f_1, \ldots, l_n : f_n$ labels l_i and formulas $f_i, 1 \leq i \leq n$ projected from words $w_i, \ldots, w_n$:



prove $\{ l_1 : f_1, \ldots, l_j : f_j \} \vdash \alpha : t$
s_1 :
1. $John : e$ use_me_last
2. $like : e \rightarrow e \rightarrow t$
3. CHOOSE $s_1 = t_i, t_i \subset t_{now}$
4. $Mary : e$ use_me_first
12. $like(Mary) : e \rightarrow t$
13. $like(John,\ Mary) : t$
exit $\alpha = like(John,\ Mary)$
LINK $v = Mary$
prove $\{ l_k : f_k, \ldots, l_n : f_n \} \vdash \beta[v/\text{Mary}] : t$
s_2 :
5. $\mathbf{u}_{he} : e$ $\Theta\mathbf{u}_{he} \notin s_2$, male($\Theta\mathbf{u}_{he}$), use_me_last
6. CHOOSE $\mathbf{u}_{he} = John$
7. $admire : e \rightarrow e \rightarrow t$
8. CHOOSE $s_2 = t_i, t_i \subset t_{now}$
9. $v : e$ use_me_first
10. $admire(v) : e \rightarrow t$
11. $admire(John,\ v) : t$
exit $\beta[v/Mary] = admire(John,\ Mary)$

Figure 1: *Utterance Interpretation in the L&R model*

is based on an adaptation of the so-called *shift-reduce* LR parsing algorithm [Aho and Johnson, 1974], but the shift-reduce actions are used merely as a guide for the L&R algorithm. Indeed, we could rationalise our implementation by saying that the shift-reduce information, determined from the grammar rules, is like associating with each word instructions like "if you've seen input X, and Y is the next lookahead, then do procedure P", for the appropriate combinations of X, Y and P.

Pitt and Cunningham [Pitt and Cunningham, 1992] describe a parser-generator, which given a grammar specification automatically generates a prolog LR parser for the language described by the grammar. The parser-generator constructs the LR parsing tables containing the shift-reduce information, and compiles each grammar rule into a Prolog clause, i.e. it turns the specification into a type of logic grammar (*cf* Definite Clause Grammars [Pereira and Warren, 1980]). The tables and the logic grammar are passed as supplements to the LR shell to form a complete LR parser for the language.

The parser-generator and parser shell are also used in our experimental implementation of L&R for a subset of English $\mathcal{E}$, where $\mathcal{E}$ is a mapping from a set of words W to <*labels, formulas, instructions*> triples, supplemented by a set of terminal categories Σ and a set of grammar rules P. The implementation comprises 3 modules:

i a language specification which includes:

- for each word $w \in W$ a predicate ς, where $\varsigma = \sigma(l : f)$ such that $\sigma \in \Sigma$ is the word's 'traditional' terminal category, and l and f are respectively the word's labels and formulas specified in the L&R lexicon;
- for each terminal category $\sigma \in \Sigma$ a Prolog clause $\sigma(X)$:- …;
- for each grammar rule $p \in P$ a Prolog clause A_p :- …;

where the bodies of the clauses $\sigma(X)$:- … and A_p:- … implement the instructions associated with the words (also as specified in the L&R lexicon);

ii a theorem prover for the logic, which applies inference rules to the data items in the linked labelled databases, such that it uses all the assumed data items in order, respects the side conditions on the data items, and succeeds if the 'last' data item is of type t;

iii an adaptation of the LR parsing algorithm (hence L&R parsing), such that for every move the following pseudo-code is executed, for i and j parser states, σ the functor of the 'lookahead predicate' ς and p a grammar rule:

```
**if** $action(\ i,\ \sigma\ ) = shift\ j$ **then** $call(\ \varsigma\ )$ **end**;
**if** $action(\ i,\ \sigma\ ) = reduce\ \ p$ **then** $call(\ A_p\ )$ **end**;
```

The interaction of the 3 modules implements the L&R algorithm.

To show this, we consider a parse of *John likes Mary, who he admires.* The implementation maintains a pointer to the current position in the current database. The lexical specifications of words contain, *inter alia*, instructions to insert the data items as assumptions at the appropriate position in the current database, and/or to put side conditions (e.g. use_me_last) on data items. Pronouns come with the instruction to CHOOSE their antecedents. In particular, the instructions associated with relative markers, such as *who*, start a new database, link this new database to the old ('parent') database, and mark the link with the last data item of type e assumed in the parent database.

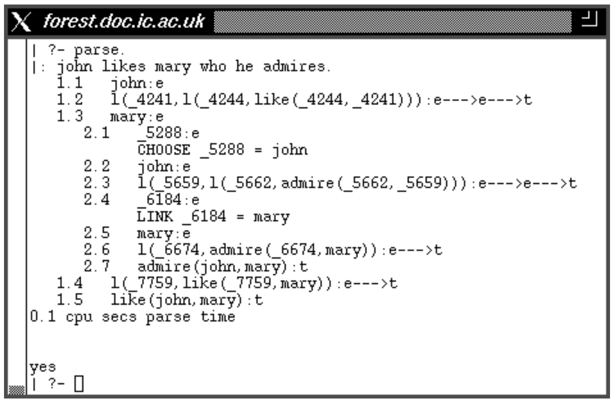
We also have the following instructions associated with the grammar rules, such that:

i reducing *via* the empty rule for noun phrases causes an assumption $v : e$ to be inserted in the current database, whereupon variable v is instantiated (LINK) to whatever marked the link between this database and its parent;

ii reducing to a sentence invokes the theorem prover, which tries to close the current database and exit with a formula $\alpha : t$;

iii reducing to a relative clause causes the pointer to be moved back to the parent database.

This process is illustrated by the automated proof/parse/interpretation shown in figure 2 below.



```
forest.doc.ic.ac.uk
| ?- parse.
|: john likes mary who he admires.
   1.1   john:e
   1.2   l(_4241,l(_4244,like(_4244,_4241))):e--->e--->t
   1.3   mary:e
      2.1   _5288:e
            CHOOSE _5288 = john
      2.2   john:e
      2.3   l(_5659,l(_5662,admire(_5662,_5659))):e--->e--->t
      2.4   _6184:e
            LINK _6184 = mary
      2.5   mary:e
      2.6   l(_6674,admire(_6674,mary)):e--->t
      2.7   admire(john,mary):t
   1.4   l(_7759,like(_7759,mary)):e--->t
   1.5   like(john,mary):t
0.1 cpu secs parse time


yes
| ?- ▯
```

Figure 2: *Automated analysis of* John likes Mary, who he admires

3.2 Example: Imperative Sentences in the Blocks World

We now work towards an account of language parsing in context to yield a full array of inferential effects. First we develop a 'blocks world', such as described by [Genesereth and Nilsson, 1987], so that it can interface with our L&R parsing mechanism. Secondly, we address the problem of the additional reasoning principles required to establish the full import of some utterances as processed in context. Finally we consider an example where the separate strands of parsing and planning can be brought together.

Essentially, the blocks world of Genesereth and Nilsson comprises:

i a description of the *objects* in the world, their *properties* and the *relations* between them: the objects include three blocks and a table, blocks may be clear, and each block may be on another block or the table;

ii a description of the permissible *actions*, which are to either stack or unstack a block, and the pre- and post- conditions on each action. These are:

Pre-conditions

To stack block x on block y:
block x must be clear,
block x must be on the table,
block y must be clear,
block x and block y must be different.

To unstack block x from block y:
block x must be clear,
block x must be on block y,
block x and block y must be different.

Post-conditions

After stacking block x on block y:
block x is on block y.

After unstacking block x on block y:
block x is on the table.

iii a description of the *axioms* which the blocks must satisfy. In this case:

each block is either on the table or on another block,
each block is either clear or has another block on it,

To describe this 'world', there are, in effect, three languages involved:

1. The database language containing the predicates $\text{on}(x, y)$ and $\text{clear}(x)$
2. The imperative (Input) command language with the predicates (actions) $\text{stack}(x, y)$.
3. The mixed metalanguage with the connectives "$\wedge$" for "and" and "$\Rightarrow$" for "precondition and action imply postcondition".

The state of the blocks world can be represented by an extensional database comprising formulas of a sorted logic, with sorts $BLOCKS = \{b1, b2, b3\}$ and $TABLE = \{table\}$, and predicates *on* and *clear*. The database can be labelled by a label representing a 'time point': performing an action then changes the state of the database (i.e. the representation of the world) and advances the time point.

The situation is illustrated by figure 3 (on page 12), which shows the database in its initial state, ready to respond to commands, and its state after executing a command to stack a block. t_0 labels all data true at the initial situation and t_1 the data true after the action. We also have:

$$t_0 < t_1,$$

and if we query the system with:

$$?\ \text{on}(b1, x)$$

we get two answers, with different labels, indicating where the answer was obtained in the database, namely:

$$\vdash t_0 :\ \text{on}(b1, table)$$
$$\vdash t_1 :\ \text{on}(b1, b2)$$

The reply to the user is determined by the system as the answer proved with the 'stronger' label, namely:

$$\text{on}(b1, b2)$$

t_0 :	t_1 :
on(b1,table)	*on(b1,b2)*
on(b2,table)	*on(b2,table)*
on(b3,table)	*on(b3,table)*
clear(b1)	*clear(b1)*
clear(b2)	
clear(b3)	*clear(b3)*

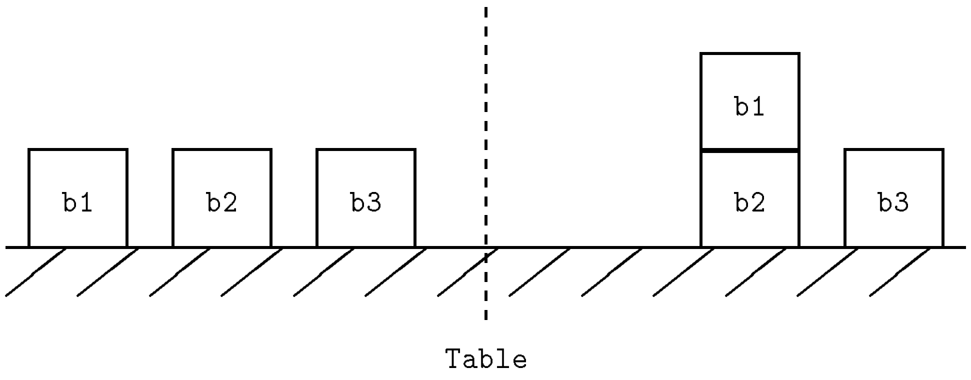


Figure 3: *Initial State and State after* stack(b1,b2)

Call the deductive system governing the planning consideration LDS_1. This system involves proving where the blocks are after which action. The system accepts commands in logical form stack(x, y), it does not accept commands in English. If the command does come in English, which we can represent as *stack x on y*, it needs to be parsed into the LDS_1 language. This is done in a parsing logic LDS_0. The schema of figure 4 below illustrates this.

$$\begin{tikzcd}
\begin{array}{c}\textbf{English Input:}\\ \text{stack } b1 \text{ on } b2\text{. move it onto the table}\end{array} \arrow[d] \\
\boxed{\text{Parsing Logic } LDS_0} \arrow[d] \\
\text{stack}(b1, b2);\ \text{move}(b1, table) \\
\boxed{\text{Planning Logic } LDS_1}
\end{tikzcd}$$

Figure 4: *Schema for parsing and planning with LDS*

The following database Δ_2 illustrates processing the database update in LDS_0. Notice that the abduction principle in LDS_0 also uses inferential effect in LDS_1. Intuitively we are trying to abduce on who "it" refers to. If we choose "it" to be a block which is already on the table, it makes no sense to move it onto the table. Thus the command when applied to the database will produce no change. The abduction principle gives preference to abduced formulas which give some effect. problem of LDS_0.

	stack:	$e \rightarrow (e \rightarrow t)$	$?S + S$
	b1:	e	
	b2:	e	
	move:	$e \rightarrow (e \rightarrow t)$	
Δ_2 :	*it*:	use abduction. First use structural abduction to get the first e higher in the list, then use inferential abduction to try and get maximal inferential effects in LDS_1.	
	table:	e	

3.3 Implementing the Blocks World

This blocks world can be implemented in Prolog, to provide a module which can be integrated with the L&R parser, comprising:

i an (extensional) database representing the state of the world;

ii procedures for executing the actions stack and unstack, and changing the state;

iii procedure for ensuring that the new state satisfies the axioms

To ensure that the labelled database satisfies the blocks world axioms, we have exploited a theorem proving technique called the *consistency method*, developed for checking integrity in *deductive* databases [Sadri and Kowalski, 1987]. For our purposes, a deductive database Δ is defined as a finite set of horn clauses of the form:

$$A \leftarrow L_1 \wedge \ldots \wedge L_n, n \geq 0$$

where A is an atom and each $L_i, 0 \leq i \leq n$, is a literal (i.e. a negated or un-negated atom). If $n = 0$ then the rule is called a *fact.* Integrity constraints $\mathcal{I}$ are a set of closed first-order formulas which Δ must satisfy, and have the form of a *denial*:

$$\leftarrow L_1 \wedge \ldots \wedge L_n, n \geq 1$$

where each L_i is a literal as before. Suppose we now have a transaction T, which in our case will involve either the addition or deletion of a fact to produce an updated database Δ_T. The consistency method uses forwards reasoning to ensure that $\Delta_T \cup \mathcal{I}$ is consistent (cf. STRIPS-style systems [Fikes and Nilsson, 1971] and its descendants).

Considering again the blocks world, the initial information state of the blocks world, encoded as an extensional labelled database, is then as shown in the left-hand side of figure 3. In (iii) we have the following (implicitly quantified) integrity constraints:

$$\mathcal{I} \;=\; \{ \quad \begin{array}{l} \leftarrow \mathit{on(X,table)} \wedge \mathit{on(X,Y)}, \\ \leftarrow \mathit{clear(X)} \wedge \mathit{on(Y,X)} \qquad \} \end{array}$$

The operation of the system incorporating the L&R parser and the blocks world module is as follows. The user enters a command, which is interpreted by the L&R parser to produce a labelled database representing its meaning. This logical expression is then executed by the blocks world module: its pre-conditions are verified as satisfiable, and the post-condition is treated as a transaction with which to update the database. The axioms are then checked, and, if necessary, the appropriate action is taken to restore integrity.

For example, in the initial state (0), from the user command *"stack block1 on block2"*, the L&R parser deduces the labelled database:

$$\begin{array}{rl} & \textbf{prove } \{\ldots\} \vdash \alpha : t \\ 0 : & \boxed{\begin{array}{l} \vdots \\ \mathit{stack(me,b1,b2)} : t \end{array}} \\ & \textbf{exit } \alpha = \mathit{stack(me,b1,b2)} \end{array}$$

(Note that here we have specified discourse referents lexically, i.e. *block1* and *block2* are treated as proper nouns referring to *b1* and *b2* respectively; similarly the system infers that the missing subject of the command is it (*me*). More rigorous treatments are possible, see e.g. [Pitt *et al.*, 1992].)

The label α is executed as a command, and the preconditions (in the initial state) are satisfied, so the database is updated with the transaction *on(b1,b2)*. Using the consistency method, the system detects violation of both integrity constraints, and infers that (one way) to restore integrity is to withdraw the facts *on(b1,table)* and *clear(b2)*. These then are the contextual implications of the command, which although no longer true in the 'current' state, are left as facts in a database labelled by the 'previous' time point, thus:

$$\begin{tikzcd}
0 : \boxed{\begin{array}{l} clear(b1) \\ clear(b2) \\ clear(b3) \\ on(b1,table) \\ on(b2,table) \\ on(b3,table) \end{array}} \arrow[r, "stack(b1{,}b2)"] & 1 : \boxed{\begin{array}{l} clear(b1) \\ clear(b3) \\ on(b1,b2) \\ on(b2,table) \\ on(b3,table) \end{array}} \\
 & 0 : \boxed{\begin{array}{l} clear(b2) \\ on(b1,table) \end{array}}
\end{tikzcd}$$

As a result, we also have a facility for 'historical' queries, and for retracting moves.

From this state, the user command *"move it onto the table"* returns the system to the same configuration as in state 0, only now labelled by 2. This is because taking the meaning of *it* to be either *b2* or *b3* gives no inferential effects, whereas *it* = *b1* does.

4 Unexpected Answers to Yes-No Questions

Our picture of dialogue, though emerging, is as yet incomplete. We have a parsing process which takes English input and delivers a predicate logic formula as the label to a conclusion in some labelled database. We have a means of capturing how this resulting formula is embedded in an update mechanism expressed in a predicate logic notation. This was only part of the interpretation problem as we face it in natural dialogue. We also have to consider the array of indirect inferential effects, utterances may give rise to before they are ever used to revise one's beliefs (or not) in some system of updating. In this latter task, we face free manipulation of abductive choices. In many cases, the participants in a dialogue reply to sentences and answer queries with deliberate indirectness. The effect is always the same: the hearer gets more information than she would by a direct answer. The paradigm case of such indirectness is the case of the indirect reply to a yes-no question. We approach this problem by considering the nature of queries in general, setting aside the parsing problem itself for the time being.

4.1 "First Principles"

Given a set of wffs Δ, taken as assumptions, and a formula or query Q, we can pose the logical question $\Delta?Q$. The meaning of this question is: Does Q follow from Δ in the logic we are dealing with, *ie* is $\Delta \vdash Q$ true? There are three possibilities:

1. $\Delta \vdash Q$ holds
2. $\Delta \vdash {\sim}Q$ holds
3. Neither $\Delta \vdash Q$ nor $\Delta \vdash {\sim}Q$ hold.

We do not adopt any additional conventions such as $\Delta \not\vdash Q$ implies $\Delta \vdash {\sim}Q$, where Q is atomic (i.e. the closed-world assumption). We thus want a definite proof of the negation ${\sim}Q$ in all cases.

We denote by $\Delta ?!Q = 1$ the situation where logically the database is consistent and is able to prove either Q or $\sim Q$. In other words $\Delta ?!Q = 1$ means that the database Δ decides the truth of Q in a definite way.

The situation we are going to study arises when there is an exchange of information between an asker, A and a responder R. At the outset of the exchange, A has a set of assumptions Δ_A: so too does R, i.e. Δ_R. Note that Δ_A may include A's beliefs about the content of Δ_R, and vice versa.

Suppose now that A asks R the query Q. If Δ_A were sufficient to prove or disprove Q then we can assume that A could have figured out Q for himself (unless, of course, he wanted to find out if R knew that Q). However, we will begin by assuming that $\Delta_A ?!Q \neq 1$.

R should now respond by supplying some new information B to be added to Δ_A to enable $\Delta_A \cup \{B\} ?!Q = 1$. The answer could be:

- $B = Q$ (i.e. "yes"), in which case $\Delta_A \cup \{Q\} ?!Q = 1$;
- $B = \sim Q$, (i.e. "no"), in which case $\Delta_A \cup \{\sim Q\} ?!Q = 1$;
- some B such that $\Delta_A \cup \{B\} ?!Q = 1$;
- some B such that $\Delta_A \cup \{B\} ?!Q \neq 1$.

In the first three cases things are fine for A. The situation becomes more interesting when the new data B is still insufficient for A to (logically) determine an answer to Q.

R may have her own reasons for not answering directly and saying Q or $\sim Q$. Perhaps she wants to forestall any future questions. Perhaps she wants to give the reason for Q (or for $\sim Q$), namely B. For whatever reason, and assuming that R is trying to be co-operative rather than evasive, A expects the answer B to logically decide the query Q, i.e. for $\Delta_A \cup \{B\} ?!Q = 1$ but nevertheless finds this not to be the case. At this point we invoke what we call *the Rule of Relevant Assumptions*; and require that further data B' should be abduced such that $\Delta_A \cup \{B, B'\} ?!Q = 1$. We assume by relevance that R gives A the minimal input B which allows him to decide the query Q. To find B' we need to know and use the computation procedure we have for $\vdash$.

We call the relevance-theoretic principle which allows us to find the enrichment B' *The Rule of Relevant Assumptions (RRA): the principle of consistent least enrichment relative to a proof procedure for* $\vdash$, whereby:

1. Given a proof procedure for finding answers to queries of the form $\Delta ?Q$, then if Δ is a partial database and we ask a query $\Delta ?Q$ and we get the input B, then we try and prove Q from $\Delta \cup \{B\}$ and assume that any information $B_1, B_2, \ldots$ required for the success of the proof was intended to be added to Δ along with the input B. $\Delta \cup \{B, B_1, B_2, \ldots\}$ must emerge as consistent. When we are talking about proof or deduction, we do not necessarily restrict ourselves to monotonic deduction. The deduction rules used in trying to prove $\Delta \cup \{B\} \vdash Q$ may be defeasible or non-monotonic. These rules correspond better to common sense reasoning. The nature of the logic involved is as yet unspecified by us.

2. If there are several ways of adding $\{B_i\}$ leading to success, we choose the one which involves the least deductive effort for some inferential effect. (These notions will be made precise in later work: they require specification of links between databases so that we can describe for example some concept of closeness between a pair of databases).

We note that the process of finding the necessary B_i requires a given proof procedure for the logic licensing the addition of such assumptions to the database. Thus for different proof procedures we may get different B_i. We also note that the asker A may get his enriched database $\Delta_A \cup \{B, B_i\}$ wrong, in which case further interaction would be required to clarify the matter.

4.2 An Example

Suppose we have a database Δ for an asker A and his query Q to a responder R. Δ could contain the fact that A has apples, and Q could be 'would you like [to eat] an apple?' Here we assume success in the parsing process is achieved and we have some outcome to be evaluated against the database, i.e.:

$$\begin{aligned} \Delta &\ni \exists x[\mathrm{Apple}(x)] \\ Q &= \exists x[\mathrm{Apple}(x) \wedge \mathrm{Eat}(x)] \end{aligned}$$

The answer A gets from the responder R may be "yes" (Q) or "no" ($\sim Q$), or A may get another answer B which allows A to conclude Q or $\sim Q$ (e.g. "I'm not hungry"). Consider instead the answer: $B =$ "I don't eat South African apples". Our proposed formal mechanism for parsing must bring the content of the sentence out, i.e.:

$$B = \forall x[\mathrm{SA}(x) \wedge \mathrm{Apple}(x) \rightarrow \sim \mathrm{Eat}(x)]$$

A has asked on the basis of Δ whether Q and got the answer B. We can therefore assume by the principle of relevance that $\Delta \cup \{B\}?!Q = 1$, i.e. the extra input information B is expected to (logically) decide the question Q. However, adding B to Δ does not decide Q in this case. Now A can invoke the principle of consistent least enrichment and try to abduce some further data B' such that $\Delta \cup \{B, B'\}?!Q = 1$.

We will now examine one way to compute the enrichment of Δ with some B' to give the desired effect. In this case, we will use as the RRA for this $\vdash$ the (propositional) abduction algorithm given by Gabbay [to appear], which we have also implemented in Prolog, whereby:

1. $\mathrm{Abduce}(\Delta, Q) = \top$ if $\Delta?Q = 1$;
2. $\mathrm{Abduce}(\Delta, q) = q$, for q atomic such that q is not the head of any clause in Δ;
3. $\mathrm{Abduce}(\Delta, Q_1 \wedge Q_2) = \mathrm{Abduce}(\Delta, Q_1) \wedge \mathrm{Abduce}(\Delta, Q_2)$;
4. $\mathrm{Abduce}(\Delta, A_1 \rightarrow (A_2 \rightarrow \ldots (A_n \rightarrow q)\ldots)) = A_1 \rightarrow (A_2 \rightarrow \ldots (A_n \rightarrow \mathrm{Abduce}(\Delta \cup \{A_1, A_2, \ldots A_n\}, q))\ldots)$;
5. Let q be atomic and let $B^j = B^j_1 \rightarrow (B^j_2 \rightarrow \ldots (B^j_{n_j} \rightarrow q)\ldots)$ for $j = 1, \ldots, m$ be all clauses in Δ with head q. Then $\mathrm{Abduce}(\Delta, q) = \bigvee_{j=1}^{m} \bigwedge_{i=1}^{n_j} \mathrm{Abduce}(\Delta, B^j_i)$.

If we introduce skolem constants and herbrand constants so that we are dealing only with propositional variables, put Δ in clausal form, and rewrite $\sim p$ as $p \rightarrow \bot$, we can apply this abduction algorithm to the example. We also focus on the relevant parts of Δ, i.e. $\exists x[\mathrm{Apple}(x)]$.

We want to know whether the answer to $\Delta \cup \{B\}?Q$ is yes or no. We therefore try and prove both Q and $\sim Q$, abducing any further assumptions along the way. We have:

1.	a	(i.e. Δ skolemised)
2.	$a \rightarrow (sa \rightarrow (e \rightarrow \bot))$	(i.e. B herbrandised in clausal form)

We want to prove $Q = a \wedge e$ and $\sim Q = (a \wedge e) \rightarrow \bot$. We will examine each in turn.

For Q we have:

$$\begin{aligned} \mathrm{Abduce}(\Delta, a \wedge e) &= \mathrm{Abduce}(\Delta, a) \wedge \mathrm{Abduce}(\Delta, e) \\ &= \top \wedge \mathrm{Abduce}(\Delta, e) \\ &= \mathrm{Abduce}(\Delta, e) \end{aligned}$$

To compute $\mathrm{Abduce}(\Delta, e)$ we have two policies. Firstly, if we don't unify e with $\bot$, then the result of $\mathrm{Abduce}(\Delta, e) = e$. Since we are assuming that the RRA yields some new information B' and $Q \vdash e$ anyway, we ignore this result.

Secondly, we do let e unify with $\bot$. Then the algorithm gives:

$$\begin{aligned}\text{Abduce}(\Delta, e) &= \text{Abduce}(\Delta, a) \wedge \text{Abduce}(\Delta, sa) \wedge \text{Abduce}(\Delta, e)\\ &= sa \wedge \text{Abduce}(\Delta, e)\end{aligned}$$

We therefore need to solve the following "equations" for the unknown $x = \text{Abduce}(\Delta, e)$:

$$\Delta \cup \{x\} \vdash e$$
$$x = sa \wedge x$$
x is minimal in satisfying the two previous properties

The only solution is $x = sa \wedge e$. This, however, will yield an inconsistent database when added to Δ. Therefore the RRA 'fails' for Q.

For $\sim Q$ we have:

$$\begin{aligned}\text{Abduce}(\Delta, (a \wedge e) \to \bot) &= (a \wedge e) \to \text{Abduce}(\Delta \cup \{a, e\}, \bot)\\ &= (a \wedge e) \to sa\end{aligned}$$

Since, in effect, this came from the herbrandised query $\text{Apple}(c) \wedge \text{Eat}(c) \to \bot$ (from $\forall y[\text{Apple}(y) \wedge \text{Eat}(y) \to \bot]$, itself rewritten from $\sim \exists y[\text{Apple}(y) \wedge \text{Eat}(y)]$, i.e. $\sim Q$), we unherbrandise to give:

$$B' = \forall x[\text{Apple}(x) \wedge \text{Eat}(x) \to \text{SA}(x)]$$

which is added to A's database Δ and so yields an answer to $\sim Q$. We now have $\Delta \cup \{B, B'\}?!Q = 1$, as required.

It should be noted that although adding B' is the minimal information enrichment for success (using an implicit ordering relation such that $p \leq q \to p \to q$), it does cost in deductive effort. This additional cost must be offset by some additional inferential effect for A, the details of which are not necessarily accessible to R. In this case, one natural such additional conclusion is that R is telling A that she believes that all the apples he has to eat are South African (and is inviting A to refute that belief before she will eat one of his apples). There may also be other conclusions in respect of other fruit R wouldn't eat for the same reason or, more vaguely, of R's political beliefs. A is encouraged to derive inferences in either of these directions given R's indirect mode of expression, but the only inference he is forced to make is that which induces an answer to his question in one step (this in itself is a reflection of the cognitive cost intrinsic to relevance).

Note that the need to add B' arose directly from the proof procedure involved. We 'hit' on the need to succeed with B' and we decided to add. Different proof procedures may give slightly different results especially when considerations like cost of deduction are involved. But given the question as input, no further enrichment (such as 'If the apples are Polish, R might not eat them either') is even attempted until it is established whether the indicated enrichment itself yields a definitive answer.

4.3 An Extended Dialogue

There are many issues to clarify here, but this provides an indication of how we think such abduction resolution should be addressed. We now turn to the analysis of an extended dialogue, in which indirect answers play a prominent part. The example shows a continuing temporal interaction which can lead to different enrichments as time progresses. Furthermore, the process of utterance interpretation in the dialogue will be embedded within an overall planning framework, in which the participants have certain desired outcomes in mind and plan what they say with these outcomes in mind, but during the course of the conversation have to renegotiate the source of conflicts between these outcomes as they become manifest in the discourse scenario.

Consider a situation in which a customer wants to hire a car 'today', and has a fair idea of the type of car she wants. She approaches a salesman who, understandably, wants to maximise his income by hiring and selling the biggest and most expensive cars. The following dialogue, let us suppose, ensues:

CUSTOMER:	*I want to hire a car.*	(1)
SALESMAN:	*When for?*	(2)
CUSTOMER:	*Today.*	(3)
SALESMAN:	*Would you like to hire this Sierra?*	(4)
CUSTOMER:	*I don't drive big cars.*	(5)
SALESMAN:	*The small cars are reserved for members only.*	(6)
CUSTOMER:	*Can I join?*	(7)
SALESMAN:	*It will take a week to process the papers.*	(8a)
	Would you like to reconsider?	(8b)
CUSTOMER:	*Big cars are not easy for me to drive. I can't control them.*	(9)
SALESMAN:	*But the Sierra has power-assisted steering!*	(10)
CUSTOMER:	*So?*	(11)
SALESMAN:	*Power-assisted steering makes a car easy to control.*	(12)

The dialogue is one which requires successive steps of abduction, both 'inferential' and 'structural'. We shall analyse this dialogue closely to try and bring this out, using formal notation quite loosely. We are assuming, *inter alia*, a logic with sorts, modal operators of wants and believes, that agents have goals and plans, temporal labels on formulas which we can reason with, etc. Because of the complexity of and interaction between these phenomena, we will gloss over many details, but aim for the exposition to establish the principles involved.

At step (1) the customer C sets out her goal, the hiring of a car. At step (2) the salesman, S, seeks to pinpoint this at some particular point in the temporal flow, and at step (3), C provides S with this information. From the salesman's point of view, we then have:

$$\begin{array}{rcl} G_S & = & W_S W_C \exists x_{car}[\text{hire}(C, x_{car}) \wedge \text{profitable}(x_{car})] \\ \Delta_S & \ni & t_{now} : W_C \exists x_{car}[\text{hire}(C, x_{car})] \end{array}$$

i.e. his goal G_S is to hire out a car 'profitably', and from (1) and (3) he has added to his database Δ_S the customer's goal.

From now on, the conversation is a successive attempt by each speaker to push the conversation towards satisfaction of their own ends - S to hire out an expensive car, C to hire a small car. We see immediately the point of indirect answers in dialogue. S offers (4). Rather than reply "no", C provides an indirect answer (5) which is intended to be pre-emptive. She is maximising to S the relevance of her utterance, since the indirect answer, albeit *a priori* more costly (cognitively), obviates a whole series of similar questions.

This pattern of question and answer is similar to that discussed in section 4.2, and it is instructive to observe the effect of applying the RRA in this case.

First we assume that in Δ_S there is the 'common knowledge' that if someone can't drive a car then they won't hire it, and that there is some correlation between the size of a car and its profitability. So "formally" we now have:

$$\begin{array}{rcl} G_S & = & W_S W_C \exists x_{car}[\text{hire}(C, x_{car}) \wedge \text{profitable}(x_{car})] \\ \Delta_S & \ni & t_{now} : W_C \exists x_{car}[\text{hire}(C, x_{car})] \\ & & \forall x_{car}[\sim \text{drive}(C, x_{car}) \rightarrow \sim \text{hire}(C, x_{car})] \\ & & \forall x_{car}[\text{big}(x_{car}) \leftrightarrow \text{profitable}(x_{car})] \\ & & B_C \forall x_{car}[\text{big}(x_{car}) \rightarrow \sim \text{drive}(C, x_{car})] \\ Q & = & W_C \text{hire}(C, \text{Sierra}_{car}) \end{array}$$

Using (intuitively) the abduction algorithm of section 4.2, Abduce$(\Delta_S, Q) = Q$, which is ignored. On the other hand, Abduce$(\Delta_S, \sim Q) = B_C \text{big}(\text{Sierra}_{car})$, which is enough to prove $\sim Q$. This then is added to Δ_S.

At this point the salesman should be able to detect a refinement of the customer's goal, i.e.:

$$t_{now} : W_C \exists x_{car}[\text{hire}(C, x_{car}) \wedge \text{small}(x_{car})]$$

and that there is a conflict between the salesman's goal and the customer's goal, caused by the customer's beliefs. Rather than relaxing his own goal, (6) is an attempt by the salesman to revise the customer's belief so that she will revise her own goal and, in turn, communicate this to him.

So he tells her that the set of cars in which she is interested are for members only. In processing (6), C has to abduce *structurally* the existence of some association (formally A_{assoc}, say) to which these members belong. Making this step, she asks (7), requiring her hearer, S, in turn to abduce structurally the second argument of the relation 'join' as being this same association. In our model, we expect that S's 'parsing database' may look something like:

$$\boxed{\begin{array}{l}
\vdots \\
\textbf{prove } \{I\colon e,\ join\colon e \to e \to t\} \vdash \gamma : t \\
\boxed{\begin{array}{rl}
n\text{:} & I\colon e \text{ — abduce: set } I = C \\
n+1\text{:} & join\colon e \to e \to t \\
n+2\text{:} & \text{abduce: } ?\colon e \text{ is required to exit} \\
n+3\text{:} & \text{abduce on structure: set } ? = A_{assoc} \\
 & \text{as right sort for second argument of join}
\end{array}} \\
\textbf{exit } \gamma = \text{join}(C,\ A_{assoc}) \\
\vdots
\end{array}}$$

We would also expect, in context, that S can work out the implication of (7), namely that if C joins this association, then she will become a member (and so eligible to hire small cars).

S, presuming C able to reason that joining an association involves filling out, sending off, and processing application papers, replies *indirectly* with (8a). She is required to abduce, again structurally, that the papers so indicated are the papers relevant to her application of this presumed association, and that if she wants to join they the papers need to be processed before she becomes a member. Furthermore, the indirect answer requires 'inferential' abductive reasoning on the customer's part. "Formally", we have:

$$\begin{array}{rcl}
\Delta_C & \ni & \forall x_{car} \forall t[t : \text{member}(C, A_{assoc}) \to t : (\text{hire}(C, x_{car}) \wedge \text{small}(x_{car}))] \\
Q & = & [t_1 \leq t_2] : \exists t_1, t_2[t_1 : \text{join}(C, A_{assoc}) \to t_2 : \text{member}(C, A_{assoc})] \\
B & = & [t_3 \leq t_4, t_4 = t_5 - 1\text{week}] : \exists t_3, t_4, t_5[t_3 : \text{join}(C, A_{assoc}) \to \\
 & & \qquad t_4 : \text{process}(\text{papers}) \to t_5 : \text{member}(C, A_{assoc})]
\end{array}$$

Applying the abduction algorithm of section 4.2 to an intuitively skolemised/herbrandised version of the above (and assuming the ability to propagate temporal constraints (i.e. labels of the form $[t_1 \leq t_2]$)), we get:

$$\begin{array}{l}
\text{Abduce}(\Delta, [t_1 \leq t_2] : t_1 : join \to t_2 : member) \\
\quad = \quad [t_1 \leq t_2] : t_1 : join \to \text{Abduce}(\Delta \cup t_1 : join, t_2 : member) \\
\quad = \quad [t_1 \leq t_2, t_2 = t_5, t_3 \leq t_4, t_4 = t_5 - 1\text{week}] : t_1 : join \to \\
\qquad\qquad \bigwedge(\text{Abduce}(\Delta \cup t_1 : join, t_3 : join), \text{Abduce}(\Delta \cup t_1 : join, t_4 : process)) \\
\quad = \quad [t_1 \leq t_2, t_2 = t_5, t_3 \leq t_4, t_4 = t_5 - 1\text{week}, t_1 = t_3] : t_1 : join \to (\top \wedge t_4 : process) \\
\quad = \quad [t_1 \leq t_4] : t_1 : join \to t_4 : process
\end{array}$$

Although the RRA has succeeded, C is also able to reason that by adding this to the database and asking herself $\Delta_C ? t_{now} : \text{join}(C, A_{assoc})$ she can conclude:

$$\forall x_{car}[[t_{now} \leq t - 1\text{week}] : t : (\text{hire}(C, x_{car}) \wedge \text{small}(x_{car}))]$$

but the time constraint on when she can hire a small car, requiring this time t to be greater than t_{now} by at least one week, renders this conclusion inconsistent with her goal.

This, according to relevance, is what the salesman intends her to abduce, and since he knows that C cannot reconcile her goal with the given information he continues with (8b). With this question, he reminds her of the first steps in their negotiation (indeed we posit that the lexical content of reconsider forces her to set up as its second argument some step in the 'consideration' so far to hand). In (9) C now provides the reason for her earlier judgment (5) and in so doing gives S the information he needs to subvert her belief and to fulfill his own goal. He responds with (10). The keyword here is *but*. We intend the procedural specification of *but* to be formulated so that it instigates a search for a set of premises that derive inconsistency at some level of inference - the inference to be derived from the second conjunct to replace the inference to be derived from the first. Since in (9) it is the customer's assertion that big cars are not safe which is the reason for her belief of (5) it is this assertion that S will want her to retract. Accordingly he utters (10) requiring her to recover premises that combine with this to replace the inference from C's utterance (9).

Despite recognising this, the customer refuses to provide the requisite premise herself. However, the word *so* provides the salesman with an invitation to build just this premise for her. We can specify *so* to be an anaphoric expression precisely for this purpose. S duly provides C with the premise needed. The customer now has to reason (non-monotonically) from (12) to the point where she can relax her goal and inform the salesman that she is prepared to hire a big car, presuming it has power-assisted steering.

This example displays all the phenomena we have so far investigated - the general parsing problem, the manipulation of abduction principles both structural and inferential, the use of words as procedural instructions dictating the flow of inference, the embedding of all such reasoning tasks in an overall cooperative reasoning enterprise which constitutes the dialogue itself. However, we are still far from formally modelling and implementing every aspect of this.

5 Conclusions

Our aim in this paper was to explore a more non-standard interaction between a human and a database, and in particular the prerequisites for equipping the database with the 'intelligence' to participate 'rationally' in natural language question-answering and dialogue. We observed that (at least) two components were necessary: a front-end module to parse natural language statements/queries and a second module for reasoning with the result of parsing. In sections 3 and 4 respectively, we described our experimental implementation of the first module and discussed (some of) the reasoning requirements of the second module. Our proposals have synthesized aspects of abductive reasoning and relevance theory in the framework of Labelled Deductive Systems, used not only to bring together aspects of the parsing and interpretation processes, but also to provide a general framework for reasoning about the implicit content of communicative acts.

Due to the breadth and complexity of the phenomena we have investigated and are attempting to model, many of the details are omitted in this paper, and some remain to be addressed in depth at all. Work remaining to be done includes:

- formalization of the trade-off between inferential effect vs. deductive effort;
- integration of the parsing system with first order abduction algorithms, set in a general knowledge representation framework;
- interaction of the results of abductive reasoning with algorithms used to provide co-operative answering from computerised databases;
- development and integration of abduction algorithm(s) for the first order case, and exploration of the space of different algorithms combined with different $\vdash$.
- futher analysis of, and integration of reasoning methods for, other 'common sense' intuitions about time, permission, beliefs, wants, etc., cf. [Ohlbach and Pratt, 1993].

We remain some way from our stated aim of 'intelligent databases', nonetheless we believe that the LDS framework provides a requisite foundation within which the heterogeneous reasoning activities involved in utterance interpertation can be set out, linked and absorbed into a single composite reasoning system.

Acknowledgements
We would particularly like to thank Robert Demolombe and two anonymous reviewers for their insightful comments on an earlier version of this paper.

References

[Aho and Johnson, 1974] A. Aho and S. Johnson. LR parsing. *Computing Surveys*, 6(2):99–124, 1974.

[Cunningham *et al.*, 1991] J. Cunningham, D. Gabbay, and H.-J. Ohlbach. Towards the MEDLAR framework. In *Esprit '91: Proceedings of the Annual Esprit Conference*, pages 822–841. Kluwer/CEC DGXIII, 1991.

[Fikes and Nilsson, 1971] R. Fikes and N. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 3(4):251–288, 1971.

[Gabbay and Kempson, 1992] D. Gabbay and R. Kempson. Natural language content: a proof-theoretic perspective. In *Proceedings of the 8th Amsterdam Colloquium on Formal Semantics*, 1992.

[Gabbay, 1990] D. Gabbay. Labelled Deductive Systems, Part I. Technical Report CIS-Bericht-90-22, Centrum für Informations und Sprachverarbeitung, Universität München, 1990.

[Gabbay, 1991] D. Gabbay. Abduction in Labelled Deductive Systems: a conceptual abstract. In R. Kruse and P. Siegel, editors, *Proceedings of the European Conference on Symbolic and Quantitative Approaches for Uncertainty*, volume 548 of *LNCS*. Springer Verlag, 1991.

[Genesereth and Nilsson, 1987] M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.

[Morrill, 1990] G. Morrill. Grammar and logical types. In *Proceedings of the 7th Amsterdam Colloquium on Formal Semantics*, 1990.

[Oehrle *et al.*, 1988] R. Oehrle, E. Bach, and D. Wheeler (editors). *Categorial Grammars and Natural Language Structures*. Reidel, 1988.

[Ohlbach and Pratt, 1993] H.-J. Ohlbach and I. Pratt. Sample dialogues. Medlar II PPR 1, Deliverable I.1.1; CEC Esprit Basic Research Action Medlar II (Esprit 6471), 1993.

[Pereira and Warren, 1980] F. Pereira and D. Warren. Definite clause grammars for language analysis. *Artificial Intelligence*, 13:231–278, 1980.

[Pereira and Warren, 1983] F. Pereira and D. Warren. Parsing as deduction. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*, Cambridge, MA, 1983.

[Pitt and Cunningham, 1992] J. Pitt and J. Cunningham. Grammars, deductive systems, and parsing. Invited talk at the *ESF Conference on Logic, Language and Information*, Autrans, France, 1992.

[Pitt *et al.*, 1992] J. Pitt, J. Cunningham, and K. Van den Bergh. An automated reasoning approach to natural language understanding. *Discipline Filosofiche (Analitika)*, 2, 1992.

[Sadri and Kowalski, 1987] F. Sadri and R. Kowalski. An application of general purpose theorem proving to database integrity. In J. Minker, editor, *Proceedings of the Workshop on Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, 1987.

[Sperber and Wilson, 1986] D. Sperber and D. Wilson. *Relevance: Communication and Cognition.* Blackwell, 1986.