Deep-reasoning-centred Dialogue

Debora Field
Department of Computer Science,
University of Liverpool, L69 3BX.

Allan Ramsay
Department of Computer Science,
University of Manchester, M60 1QD.

Abstract

This paper discusses an implemented dialogue system which generates the meanings of utterances by taking into account: the surface mood of the user's last utterance; the meanings of all the user's utterances from the current discourse; the system's expert knowledge; and the system's beliefs about the current situation arising from the discourse (including its beliefs about the user and her beliefs, and its beliefs about what is 'common knowledge'). The system formulates the content of its responses by employing an epistemic theorem prover to do deep reasoning. During the reasoning process, it remembers the proof tree it constructs, and from this derives the meaning of an explanatory response.

1 Introduction

We are building a system that offers users expert advice (on health, but the particular domain is unimportant) by means of multiple-turn natural language dialogue. The work presented in this paper concerns largely the generation of the *meanings* of utterances, meanings which contain all the information necessary for their realisation in grammatical surface form. (The surface realiser is as yet unfinished, but we do not envisage many problems in completing it, as will be discussed.) In answer to the old chestnut, "Generation from what?", the system generates the meanings of its utterances by analysing and manipulating a range of different sorts of information, including: the surface mood of the user's last utterance; the meanings of all the user's utterances from the current discourse; the system's expert knowledge about health; and the system's beliefs about the current situation arising from the discourse (including its beliefs about the user and her beliefs, and its beliefs about what is 'common knowledge'). To describe what the system is generating and how it does this therefore requires description of the entire process from analysis of the surface structure of a user's utterance, right through to the point at which an utterance meaning is finished and ready to be sent to the surface realiser. We will start with a brief outline of this process, which will be expanded upon in the body of the paper.

The anticipated user is a person wanting health advice, who engages the system in a turn-by-turn NL dialogue by typing in utterances in grammatical English, requesting responses from the system, and responding to them. The system understands users' utterances, in the sense that for each utterance it derives a meaning expressed in intensional logic. As the conversation proceeds, the system also maintains an understanding of the discourse as a whole, in that after each utterance it updates a growing model of the discourse, enabling it to remember everything that has been said during the discourse, who said what, which order the utterances were made in, which entities have been mentioned during the discourse (and during which turn), and how they were referred to.

The system has its own belief state—including its beliefs about what the user believes—which it continually updates, as it keeps track of the meanings of what has been said by whom during the dialogue, and as it refers to its bank of 'common' knowledge. The system also has a bank of expert knowledge on health, expressed as rules which describe causal relations between conditions, foods, activities, *etc.*, which enable it to reason about the consequences of particular choices and actions.

The system's epistemic theorem prover does deep reasoning over its understanding of and beliefs about what the user has said (expressed as logical forms), its expert knowledge, and its memory of the contents of the whole discourse, in order to formulate accurate and appropriate response meanings during dialogue with users. The system also enhances its responses by making them explanatory, which it does by using the edited contents of the proof tree that gets constructed during the first stages of calculating the system's response to an utterance.

Having formulated an utterance meaning that it believes will be accurate and appropriately explanatory, the system's belief state and discourse model are accordingly updated, and the system fleshes out the utterance meaning by deriving descriptions for all the nominal entities in the utterance meaning, so that the user will be able to dereference them, and so that natural-sounding discourse results. Then the system maps the fleshed-out utterance meaning into grammatical surface form. The system's surface realiser is unfinished, however, we are confident that the nature of the input to the realiser is what is needed for our system to produce grammatical surface text, and that we have a suitable architecture for realising it.

2 NL understanding

We can view our system as an utterance generator whose input is user utterances. On receiving a user's utterance, the system derives a logical representation of its meaning (Ramsay, 2001b; Ramsay, 1997):

(1) U: *I am allergic to eggs.*

Figure 1: Logical form of (1) after analysis: 30 ms

```
utt(claim,
 exists(B :: {aspect(now,simple,B)},
  state(B)
  & theta(B,pred,lambda(C,allergic(C)))
   & theta(B,topic(ref),ref(lambda(D,speaker(D)))!0)
    & lambda(E,to(B,E)) < lambda(F,egg(F))))
```

Fig. 1 shows the logical form of (1) after analysis. While the system understands that Fig. 1 is the meaning of the user's utterance, it also has the ability to reason about what follows from this, because it has its own banks of common sense knowledge and expert knowledge. The expert knowledge is expressed in the form of rules which describe causal relations between conditions, foods, activities, *etc.*, which enable the system to reason about and discuss the consequences of particular choices and actions. For example, it knows that if a person has an allergy to a substance, the allergy will be to some set of things that contain that substance, that it is dangerous for that person to eat things that belong to that set, and that something cannot be both safe and dangerous (Fig. 2). If the user follows (1) with:

(2) U: *Is it safe for me to eat pancakes?*

the system is able to use its knowledge about the consequences of having an allergy, and about the ingredients of pancakes, plus one or two other meaning postulates, to work out that it is not safe for the user to eat pancakes. What is more, the system uses the knowledge it discovers (while formulating its response) to construct an explanatory utterance which answers the user's question *and* gives justification for the answer (see section 7).

Note that the implication in the final rule of Fig. 2 is between two situation types (denoted by propositions). Reasoning about situation types seems to be essential for the correct analysis of terms like 'safe' and 'dangerous', and so we are forced to use a fine-grained intensional logic as our basic framework in

Figure 2: Some of the system's beliefs

```
bel(system,
forall(A,
 forall(O :: {allergy(O, A)},
  forall(H :: {have(H) & theta(H, object, O)},
   forall(S :: {theta(H, agent, S)},
    allergic(S, A)))))
&
forall(S :: {theta(S, pred, lambda(X, allergic(X)))},
 forall(T :: {theta(S, topic(ref), T)},
  forall(A :: {subset(lambda(X, to(S, X)), A)},
   allergic(T, A))))
&
forall(A,
 forall(S1 :: {allergic(A, S1)},
  dangerous(exists(EAT,
   eat(EAT)
   & theta(EAT, agent, A)
   & exists(X, theta(EAT, object, X)
    & exists(Y, contains(X, Y) & (S1:Y)))))))
&
forall(SOA1,
 forall(SOA2,
  not(safe(SOA1)
   & dangerous(SOA2)
   & (SOA1 => SOA2)))))
```

the same way that situation semantics is grounded in (Aczel, 1988)'s notion of non-well-founded sets. Our theorem prover (Ramsay, 2001a) allows us to perform inference over intensional rules of this kind.

3 Mood and extra-linguistic plans

Notice from Fig. 1 that the logical form of the meaning of (1) is nested inside the expression `utt(claim...)`. This arises from analysis of the surface mood of the utterance. We consider that linguistic actions are generally intended to help with underlying extra-linguistic plans, so we analyse the surface mood to determine the answer to the most basic goal-related question: What does the user want? Does she want to know whether something is true? Does she want to know the identity of something? Does she want the system to do something? We take it that: for a statement P there is some action A which the hearer ('H') could do if H knew that the propositional content P was true; for a query P there is something the speaker ('S') could do if S knew that P was true; and for an imperative P there is something that someone (probably S or H, but not necessarily) could do if H carried out the action described by P.

4 Discourse model

Having derived the meaning of a user's utterance, the system adds the meaning plus a record of who uttered it to the 'minutes' (after (Thomason, 1990; Lewis, 1979; Stalnaker, 1972)). Individual entities (including the events) that have been mentioned during the discourse are represented in a 'discourse model' as unique skolem constants, while n-place predicates describe the detail of what has been said about each entity (Ramsay and Seville, 2000). The skolems are anchored (Barwise and Perry, 1983) and note is taken of the order of centres (Grosz et al., 1995) so that NPs in later user utterances can be dereferenced successfully, and that in its own utterances, the system will know how to refer to entities in ways which enable the user to dereference them.

The discourse model enables the system to remember everything that has been said during the discourse, who said what, which order the utterances were made in, which entities have been mentioned during the discourse (and on which turn), and how they were referred to. This enables the system, during the formulation of a response to a user, to use information that she introduced into the conversation during turns preceding the most recent one, or to use information that she has introduced over a number of separate turns, as in this example ('C' is 'computer'):

(3) U: *I am allergic to eggs.*
C: *Please go on.*
U: *Is it safe for me eat pancakes?*
C: *No, it is not safe for you to eat pancakes.*
U: *What about Yorkshire pudding?*
C: *No, it is not safe for you to eat Yorkshire pudding either.*

5 System's private beliefs

The system has its own set of beliefs. These include 'common knowledge' beliefs (things the system believes everyone can be expected to know), beliefs about the system's current situation arising from the dialogue (including beliefs about what the user believes), and expert knowledge about health. Having understood the meaning of a user's utterance, and updated the minutes, the system adjusts its private beliefs about the current situation in light of what has just been said. (Note that the minutes do not contain what the user believes or what the system believes concerning what has been said, it is purely a record of what has been said, and by whom.)

In adjusting its private beliefs following a user utterance, the system makes a significant default assumption. We consider that, in a purely neutral context where neither party has any specific views on the reliability or cooperativeness of the other, it is rational for a speaker to produce utterances that she believes, and for the hearer to believe this is what the speaker is doing. This default assumption, that people are committed to what they say, arises as a consequence of our assumption that linguistic actions are generally intended to help with underlying extra-linguistic plans. Since I cannot be expected to help you unless I know something about your plan, there is no point in you telling me about things that will not help me identify your plan.

Under the assumption that people are committed to what they say, during its update step following a user's utterance, the system assumes that the user has been honest in the declaration of her extra-linguistic plan (Section 3), and if the user has made a statement, it also adds to its beliefs that it believes: (i) what the user has stated; (ii) the user believes what she has stated.

The assumption we make that people are committed to what they say ignores the fact that in normal human-to-human conversation, people often say things that they themselves do not believe (lying, bluffing, or using sarcasm, for example). Enabling a dialogue system to handle conversations in which users say things they do not believe is work we have in mind for the long-term future (after (Field and Ramsay, 2004)). It is not an ability we consider essential for the proper functioning of a dialogue system whose role is to advise users who willingly approach with a genuinely enquiring attitude, in contrast to users who are perhaps obliged to use a 'counselling system' (as part of a compliance programme, perhaps), and as a consequence may attempt to deceive or be uncooperative in other ways.

6 Response strategy

Surface mood gives a strong starting point for how the system will go about working out a suitable response to a user's utterance. If the utterance is a polar query, for example:

(4) U: *Is walking good for me?*

Figure 3: Logical form of (4) after analysis: 20 ms

```
utt(query,
 exists(B
 ::{aspect(now,simple,B)},
  state(B)
  & theta(B,pred,lambda(C,good(C)))
   & lambda(D,theta(B,topic(ref),D))
    < lambda(E,exists(F,walk(F) & theta(F,agent,E)))
    & for(B,ref(lambda(G,speaker(G)))!4)))
```

the system's analyses its mood as *query* (Fig. 3) and then the system's theorem prover reasons about the proposition PROP nested inside utt(query,PROP). It first tries to prove PROP is true. If this fails, it tries to prove that PROP is false. If both proofs fail, the system has nothing concrete to report to the user.

If the user's utterance was a WH query:

(5) U: *Which foods contain eggs?*

the NL understander analyses its mood as *whquery* (Fig. 4), and then the theorem prover tries to find a proof that there is something that satisfies the given property, *i.e.*, tries to find a value of B which makes the embedded proposition true.

Figure 4: Logical form of (5) after analysis: 10 ms

```
utt(whquery,
 lambda(B,
  exists(C :: {aspect(now,simple,C)},
   contain(C)
   & lambda(D,theta(C,object,D)) < lambda(E,egg(E))
    & theta(C,agent,B))
  & food(B)))
```

If the user's utterance was a statement, the only implemented strategy the system currently has is to say something polite and banal to indicate that it is listening, as in:

(6) U: *I am overweight.*
C: *Please go on.*
U: *Is walking good for me?*

Since the system remembers everything that has been said during a conversation, if it is presented with a statement it is not sure what to do with, it is able to wait and see whether the user adds something to the discourse that makes her goal clearer.

There is a lot of work to be done to enable the system to respond more intelligently to statements, but nothing that cannot be done by approaching utterances as declarations of complex goals requiring recognition by the system, and by having plenty of meaning postulates (and an appropriate theorem prover) which enable the system to understand what follows from users' utterances. Immediate plans include treating discourse markers in user utterances as clues to the user's goal. Another is trying to judge whether the statement might be a comment on an earlier utterance from the current discourse, to analyse what kind of comment it is (dissatisfaction with, surprise at, anger at an earlier utterance, *etc.*), and to decide what would be an appropriate response.

We have not yet given much thought to how the system should respond to utterances in the imperative mood, since it is an unlikely occurrence in the context of a dialogue between a human non-expert and a disembodied machine expert that has no ability to do anything but communicate through text, apart from instances where the user explicitly instructs the system to communicate things to her (*Tell me ...*, *Explain why...*, *List...*, *etc.*), and perhaps instances where the user is insulting the system (*Get lost!*, *Go and learn English!*, *etc.*).

7 Generating explanatory responses

Let us assume the system has reached a point at which it has formulated the meaning of an accurate and relevant response to the user's utterance. We want the system to give a reply to the user which not only satisfies her communicative goal, but which satisfies it in a way that is as helpful as possible to her. With respect to WH queries to which there are several, or even very many possible answers, the issue of the relevance or helpfulness of different ways of responding has been discussed at length (see (Groenendijk and Stokhof, 1997)). If a user asks:

(7) U: *Which foods should a person with acne avoid?*

there may be hundreds or even thousands of foods which enable the proof of *A person with acne should avoid eating [what]* to be completed. We consider (following (Ginzburg, 1996)) that a descriptive or 'explanatory' answer is more useful or appropriate to the user than a list of instances, in cases where there is more than a handful of instances. By 'explanatory answer', we mean that, where possible, responses by the system should inform the user about causal relations between entities, so that she will be able to re-use her new information in other situations by making new inferences of her own. For example, rather than making the following accurate but minimal responses:

(8) U: *I have acne.*
C: *OK.*
U: *Is eating chocolate good for my skin?*
C: *No.*
U: *What about bacon?*
C: *No.*
U: *What about ... No ... What about ...*

the system should produce an explanatory response which aims to educate the user:

(9) U: *I have acne.*
C: *OK.*
U: *Is eating chocolate good for my skin?*
C: *No, because chocolate is a fatty food, and fatty foods aggravate acne.*

Now the user can infer for herself that bacon, chips, mayonnaise and cream cakes are bad for acne, whereas carrots, cod and honey are not (if she knows which foods are fatty foods and which are not).

A side-effect of the fundamental design of our system—to use a theorem prover for property theory to do deep reasoning with logical forms derived from NL utterances, in conjunction with its own expert knowledge—is that during the process of formulating a response to a user, the system discovers a lot of information which it knows could be of great potential benefit to the user. If the system is privy to all this information, why waste it by keeping it secret? Why not pass it on to the user? This is far from suggesting that the system dump all its expert knowledge on the user at once, regardless of the user's desires and needs. It is viewing the user's own utterances as insights into the gaps in the user's knowledge that the user is keen to have filled, and then attempting to fill those particular gaps as precisely as possible, neither being more informative than necessary, nor less.

Technically speaking, the explanatory information contained in the 'because...' clause in (9) is drawn from the proof tree that was constructed by the theorem prover while working out its response to the user's question (after (Fiedler, 1998)).[1] The structure of all proof trees is as follows:

```
(10) g1 +[g11 +[g111 +[]
               ],
          g12 +[g121 +[],
                g122 +[g1221 +[]]
               ]
         ]
```

where `G +[LIST...]` means `[LIST...] are the subproofs of GOAL`.

Fig. 5 is the proof tree that the system constructs while it is formulating its response to utterance (11):

(11) U: *I am overweight. Is walking good for me?*

One change has been made to make this proof tree easier to read. In the tree from the implementation, the skolem function `#813(lambda(B, exists(C, walk(C) & theta(C, agent, B))))` appears many times, and means roughly *There is a set of walking events C whose agent is some individual B.* This expression has been simplified to skolem `#813`.

Figure 5: Proof tree for response to (11): 20 ms

```
answer::[bel - computer]
 + [for(#813,S)::[bel - computer]
 + [lambda(B,exists(C,walk(C) & theta(C,agent,B)))
  < lambda(D,exercise(D))::[bel - computer] + [],
  overweight(S)::[bel - computer] + []],
 lambda(E,theta(#813,topic(ref),E))
 < lambda(B,exists(C,walk(C) & theta(C,agent,B)))
  ::[bel - computer]
 + [lambda(B,exists(C,walk(C) & theta(C,agent,B)))
  < lambda(F,exercise(F))::[bel - computer] + [],
  overweight(S)::[bel - computer] + []],
 theta(#813,pred,lambda(G,good(G)))::[bel - computer]
 + [lambda(B,exists(C,walk(C) & theta(C,agent,B)))
  < lambda(H,exercise(H))::[bel - computer] + [],
  overweight(S)::[bel - computer] + []],
 state(#813)::[bel - computer]
 + [lambda(B,exists(C,walk(C) & theta(C,agent,B)))
  < lambda(I,exercise(I))::[bel - computer] + [],
  overweight(S)::[bel - computer] + []],
 aspect(now,simple,#813)::[bel - computer]
 + [lambda(B,exists(C,walk(C) & theta(C,agent,B)))
  < lambda(J,exercise(J))::[bel - computer] + [],
  overweight(S)::[bel - computer] + []]]
```

In the construction `P::[bel-computer]` (from Fig. 5), `[bel-computer]` is the 'epistemic context' of `P`. `P::[bel-computer]` means *The system believes P* (see Section 7.1). You will notice that the epistemic context of *every* proposition in Fig. 5 is `[bel-computer]`. The system has, however, used beliefs with different epistemic contexts to construct this tree. For example, since the user has just told the system she is overweight, the fact that she is overweight is in the common ground (is assumed to be mutually believed by system and user) before the system starts to formulate its response and derive the proof tree. We do not see common-ground contexts in the proof tree, because the system's goal was to prove that it *privately* believed something, and when reasoning with common-ground propositions, the system knew that `P::[bel-commonground([user,computer])]` subsumes `P::[bel-computer]`. Fig. 6 is a paraphrase of Fig. 5 from which the epistemic context `[bel - computer]` has been removed.

Figure 6: Paraphrase of (5)

```
answer::[bel-c]
 + [(#813 is something which is 'for' S)
   + [(the set of walking events is a
       subset of type 'exercise')
      + [],
      (S is overweight) + []],
   (the set of topics of #813 is a
    subset of the set of walking events)
   + [(the set of walking events is a
       subset of type 'exercise')
      + [],
      (S is overweight) + []],
   (The predicate of #813 is type 'good')
   + [(the set of walking events is a
       subset of type 'exercise')
      + [],
      (S is overweight) + []],
   (#813 is a state)
   + [(the set of walking events is a
       subset of type 'exercise')
      + [],
      (S is overweight) + []],
   (The tense and aspect of #813 are
    'present' and 'simple')
   + [(the set of walking events is a
       subset of type 'exercise')
      + [],
      (S is overweight) + []]]
```

7.1 Explanatory responses: difficulties

In line with (Grice, 1975), we consider that a guiding principle when it comes to formulating natural responses to users' utterances is to not be overly informative, which means the user's knowledge must be taken into account (Cawsey, 1990; Paris, 1991). If a user has said the following:

(12) U: *I have acne. Is eating chocolate good for my skin?*

we think that most people would consider the following explanatory response to be odd, even though

[1]This is achieved by using an abbreviated copy of the proof stack in a 'label' (after (Gabbay, 1996)). The label carries non-logical, arbitrary information about the progress of a proof, and is used for a variety of purposes in addition to keeping a note of the proof trail. Labels are threaded through the clause, so that information can be passed from one subgoal to the next.

all of its parts are necessary in constructing the system's proof of the answer "No":

(13) C: *No, because you have acne, and eating fatty foods aggravates acne, and chocolate is a fatty food.*

The system knows (from (12)) that the user knows she has acne, so it is odd-sounding for the system to say *because you have acne* to the user, even though this proposition forms a requisite part of the system's own proof that chocolate is bad for the user's skin. In order to prevent the system being more informative than required, we eliminate from the explanatory bit of the response the parts of the proof tree that the system believes the user already knows.

Technically speaking, this is achievable on account of a significant design feature of the theorem prover, namely that it is an epistemic one (see (Field and Ramsay, 2006)). We take the view that the best way to reason about what someone else believes is to see which conclusions you would draw if your view of the world matched theirs, and to express this, we use the notion of the 'epistemic context' in which a proposition is available. We write `P::C` to say that the proposition `P` is available in the context `C` (after (Wallen, 1987)), and we let belief statements introduce contexts. Each proposition in the proof tree from which explanatory answers are derived has its own epistemic context. This information, along with the contents of the discourse model, and axioms defining the properties of knowledge and belief, enable the system to establish which parts of the proof tree it thinks are things that are already known by the user, and which parts are not already known.

A harder problem than avoiding being too informative in an explanatory response is the problem of spotting the occasions when it is not appropriate at all to give an explanatory response. Consider:

(14) U: *I am allergic to eggs. Is it dangerous for me to eat pancakes?*
C: *Yes, because if you are allergic to a substance, it is dangerous to eat foods containing that substance, and pancakes contain egg.*

We would argue that most Western readers would judge from (14) that the user probably already knows that the consequences of having an allergy to a substance are that it is dangerous to eat foods containing that substance. So it seems overly informative and even patronising for the system to explain these consequences to the user. We would also argue that, although she has not declared it explicitly, the thing the user wants to achieve in making this utterance is to find out whether pancakes contain egg, and that an accurate and minimal "Yes" would be the ideal system response. The system's problem is, it is difficult to spot when it is appropriate to assume that a user understands a relationship that she has not explicitly declared. Making decisions like this would require judgements about the inferences a particular user is likely to be able to make (Horacek, 1997; Zukerman and McConachy, 1993)—just one of the many refinements in the queue.

8 From meaning to message skeleton

In technical terms, the content of the system's proof tree is the bulk of the meaning that becomes the input to the surface generator. Formulae in proof trees are an alternative representation of the logical forms that the system has been reasoning with: the logical forms have been skolemised and anchored to make them more suitable as input for NL generation.

Let us return to user query (11), and let us assume that the system wants to use the full proof tree (Fig. 5) in an explanatory response to the user. To express Fig. 5 to the user in a natural-sounding way, the system makes an utterance with the structure *A because B*, where *A* constitutes a response that would stand alone in being accurate and appropriate, and *B* constitutes additional explanatory information. The part of the proof tree that provides the content for *A* is the list of the 'principal subgoals', which in (10) are `[g11,g12]`. The part of the proof tree that provides the content for part *B* of the response *A because B* is a list of the 'secondary subgoals', items that are nested inside the top-level subgoals, which in (10) are `[g111, g121, g122, g1221]`. Parts *A* and *B* are then somewhat crudely glued together with a `because`, to make what we are calling a 'message skeleton', which is the skolemised and anchored meaning (of the system's utterance) that will soon become the input to the surface realiser (illustrative figure coming shortly).

9 Fleshed-out message skeletons

Before the message skeleton is sent to the surface realiser, work is done to decide how to refer to nominal entities in such a way as is natural-sounding, and will enable the user to dereference them. This involves determining whether entities have been mentioned in the discourse thus far, or are in the common ground for some other reason, and if so, how they are described. An entity which is not yet in the common ground needs slightly different treatment—the system examines what it knows about that entity, and uses that information to look in a 'reverse dictionary' (in which entries are meanings, and definitions are words) for a suitable word or phrase to describe it. Additionally, after (Dale, 1988; Reiter, 1990), if an entity has not been mentioned before, it is marked for the realiser as *indefinite*. If it has been mentioned before, and can be realised by a pronoun without being misleading (*e.g.,* due to gender confusion), it is marked as *pronoun*. Otherwise, entities are marked as *definite* and minimal distinguishing descriptions obtained for them.

Fig. 7 (actual output) shows the fleshed-out message skeleton for the system's response to (11). `BECAUSE` has been added to the message, as discussed, to separate the explanatory part from the remainder in a way that reflects surface order. `YES` has been added because the system *has* constructed a proof of the proposition the user is querying, and so some surface affirmative is required. Of course, `YES` does not appear in all system utterances, since sometimes the system wants to say `NO`, sometimes it is answering a WH query, and sometimes it is making a statement. Notice the item `PRON(S,lambda(N,user(N)))`. This instructs the realiser that `S` is the user, and to realise `S` with a pronominal form. `PRON(S,lambda(N,user(N)))` was added to the bare message skeleton during the fleshing-out process. The realiser knows that the pronoun to use to refer to the user while talking to her is *you*.

It is probably unwise to show what the surface realiser outputs, given Fig. 9 as input, because it is unfinished, and its output is odd and incomplete. However, for the sake of evaluation, here it is, the time taken to produce it (from being asked the question by the user) being 590 ms:

(15) C: *[Yes, ????, because, you, be, overweight]*

This messy output should and will soon be:

(16) C: *Yes, walking is good for you, because you are overweight, and walking is a type of exercise.*

How natural this response is, and whether it should contain more information or less information, is under on-going scrutiny, as is the exploitation of the proof trees in general.

10 Conclusion

We consider the surface realiser to be the main weakness in the current implementation, but believe this should not present too many difficulties, since we already have one of the most important ingredients of a surface realiser: comprehensive input that contains all the information necessary for the generation of natural-sounding discourse turns. Also helping us is that we have resources available to us in the architecture of the system, namely, those that are used in the analysis of surface form during NL understanding. A reverse dictionary is also already in place, which is being used to flesh out message skeletons with appropriate referring and indefinite expressions for skolem constants. We intend to exploit previous work on bag generation to help finish off the realiser.

Concerning evaluation, we have included timings in the figure captions, and one timing of 590 ms for the system's complete response to (11). Its response to (2) takes 451 ms. These timings are slightly lower than they will be when the realiser is finished.

We are at pains to point out that the system is not a health expert, and the expert rules it uses are not rules that actual doctors and nutritionists use—they are rules that we have made up, and they certainly are not fine-grained or precise enough for the health domain. It is our priority to get the basic architecture and functioning complete before the system would be suitable for developing into an actual health advisor. Having said that, we are shortly about to incorporate a nutrition ontology into the system, to enable it to answer detailed questions about which foods contain which substances, and which foods should be avoided or promoted under various conditions. We are interested to see how difficult it would be for us to exploit an ontology of expert knowledge for our dialogue purposes.

The proof trees that the system's theorem prover constructs provide us with much food for thought, and could be used in ways not discussed in the paper. One use would be to generate dialogue clarification questions (after (Cawsey, 1991)). Instead of using the full proof tree, the system could cherry pick from the proof tree, and query the user on her knowledge of the different parts of the proof, with the aim of finding where the user's reasoning had gone wrong, and correcting it.

With regard to answering questions to which there is more than a handful of answers, we acknowledge that there are many occasions when using the content of a proof tree as the substance of the system's answer, with a few odd words crudely inserted, will not be satisfactory. For example, if a user said:

(17) U: *I have heart disease. What lifestyle advice can you give me?*

the natural response would be a piece of prose in which several different kinds of lifestyle change were discussed. To do this, the system would require the ability to write an appropriate summarised text on the basis of its expert knowledge. This would require the additional knowledge of how to construct an argument, as well as more in-depth knowledge of discourse structure (unless stock answers to anticipated questions had been prepared manually in advance for such occasions), work we consider feasible, given collaboration with argumentation experts, and considerable further development.

We acknowledge that our work is just one of many in a tradition of dialogue system implementations, (TRAINS (Allen et al., 1996), TRINDIKIT (Larsson and Traum, 2000), and GoDiS (Larsson et al., 2000), to mention a few).

11 Acknowledgements

Our thanks go to Floriana Grasso for her valuable input. This work was funded under EU-grant

Figure 7: Fleshed-out message skeleton in response to (11)

```
[YES,
 aspect(now,simple,#813(lambda(B,exists(C,walk(C) & theta(C,agent,B))),S)),
 state(#813(lambda(D,exists(E,walk(E) & theta(E,agent,D))),S)),
 theta(#813(lambda(F,exists(G,walk(G) & theta(G,agent,F))),S),pred,lambda(H,good(H))),
 lambda(I,theta(#813(lambda(J,exists(K,walk(K) & theta(K,agent,J))),S),topic(ref),I))
 < lambda(J,exists(K,walk(K) & theta(K,agent,J))),
 for(#813(lambda(L,exists(M,walk(M) & theta(M,agent,L))), S), S),
 PRON(S,lambda(N,user(N))),
 lambda(H,good(H)),
 BECAUSE,
 overweight(S),
 lambda(O,exists(P,walk(P) & theta(P,agent,O)))
 < lambda(Q,exercise(Q))]
```

FP6/IST No. 507019 (*PIPS*: Personalised Information Platform for Health and Life Services).

References

P Aczel. 1988. *Non-Well-Founded-Sets*. Stanford: CSLI Publications.

J. F. Allen, B. W. Miller, E. K. Ringger, and T. Sikorski. 1996. A robust system for natural spoken dialogue. In *Proc. ACL'96*.

J. Barwise and J. Perry. 1983. *Situations and Attitudes.* Cambridge, MA: Bradford Books.

A. J. Cawsey. 1990. Generating explanatory discourse. In Mellish, C., Dale, R. and Zock, M. (eds) *Current Research in Natural Language Generation*, Academic Press, pp. 75-101.

A. J. Cawsey. 1991. Generating interactive explanations. In *Proc. 9th National Conference on Artificial Intelligence (AAAI-91)*, pp. 86–91.

P. R. Cohen, J. Morgan, and M. E. Pollack. 1990. Editors. *Intentions in communication.* Cambridge, Massachusetts: MIT.

R. Dale. 1988. *Generating Referring Expressions in a Domain of Objects and Processes.* PhD Thesis, Centre for Cognitive Science, University of Edinburgh.

A. Fiedler. 1998. Macroplanning with a cognitive architecture for the adaptive explanation of proofs. In *Proc. 9th International Workshop on Natural Language Generation (INLG-98).*

D. Field and A. Ramsay. 2004. Sarcasm, deception, and stating the obvious: Planning dialogue without speech acts. *Artificial Intelligence Review* 22(2): 149–171.

D. Field and A. Ramsay. 2006. Planning ramifications: When ramifications are the norm, not the problem. In *Proc. 11th International Workshop on Non-Monotonic Reasoning (NMR'06)*, 30 May–1 June 2006, Lake District, England, pp. 343-351.

D. M. Gabbay. 1996. *Labelled deductive systems.* Oxford University Press: Oxford.

J. Ginzburg. 1996. The semantics of interrogatives. In S. Lappin (ed) *Handbook of contemporary semantic theory*, Oxford: Blackwell.

H. P. Grice. 1975. Logic and conversation. In P. Cole and J. Morgan (eds) *Syntax and semantics, Vol. 3: Speech acts*, pp. 41–58. New York: Academic Press.

J. Groenendijk and M. Stokhof. 1997. Questions. In J. van Benthem and A. ter Meulen (eds) *Handbook of Logic and Language*, Amsterdam/Cambridge, MA: Elsevier/MIT Press, pp. 1055-1124.

B. J. Grosz, A. Joshi, and S. Weinstein. 1995. Centering: a framework for modeling the local coherence of discourse. *Computational Linguistics* 21(2): 175–204.

H. Horacek. 1997. A model for adapting explanations to the user's likely inferences. *User Modeling and User-Adapted Interaction* 7(1):1–55.

S. Larsson and D. Traum. 2000. Information state and dialogue management in the TRINDI Dialogue Move Engine Toolkit. In *Natural Language Engineering Special Issue on Best Practice in Spoken Language Dialogue Systems Engineering*, CUP, UK pp. 323-340.

S. Larsson, P. Ljunglf, R. Cooper, E. Engdahl, and S. Ericsson. 2000. GoDiS - an accommodating dialogue system. In *Proc. ANLP/NAACL-2000 Workshop on Conversational Systems*, pp. 7-10.

D. Lewis. 1979. Scorekeeping in a language game. *Journal of Philosophical Logic* 8: 339–59. Reprinted in D. Lewis *Philosophical papers Volume I*, 1983, New York and Oxford: Oxford University Press, pp. 233–249.

C. Paris. 1991. The role of the user's domain knowledge in generation. *Computational Intelligence* 7:71–93.

A. Ramsay and H. Seville. 2000. Models and discourse models. *Journal of Language and Computation 1(2).*

A. M. Ramsay. 1997. Dynamic and underspecified semantics without dynamic and underspecified logic. In H. C. Bunt and L. Kievit and R. Muskens and M. Verlinden (eds) *Computing Meaning* 1: 208–220, Dordrecht: Kluwer Academic Publishers (SLAP 73).

A. M. Ramsay. 2001a. Theorem proving for untyped constructive λ-calculus: implementation and application. In the *Logic Journal of the Interest Group in Pure and Applied Logics*, Vol. 9(1): 83–100.

A. M. Ramsay. 2001b. Weak lexical semantics and multiple views. In H. C. Bunt and R. Muskens and E. G. C. Thijsse (eds) *Computing Meaning* Vol. 2: 97–112, Dordrecht: Kluwer Academic Publishers (SLAP 77).

E. Reiter. 1990. Generating descriptions that exploit a user's domain knowledge. In R. Dale, C. Mellish, and M. Zock (eds) *Current Research in Natural Language Generation.* London: Academic Press, pp. 257-285.

R. Stalnaker. 1972. Pragmatics. In D. Davidson and G. Harman. Editors. *Semantics of natural language (Synthese Library, Vol. 40)*, pp. 380–97. Dordrecht, Holland: D. Reidel.

R. H. Thomason. 1990. Accommodation, meaning, and implicature: Interdisciplinary foundations for pragmatics. In (Cohen et al., 1990), pp. 325–63.

L. Wallen. 1987. Matrix proofs for modal logics. *Proc. 10th IJCAI*, pp. 917–23.

I. Zukerman and R. McConachy. 1993. Generating concise discourse that addresses a user's inferences. In *Proc. IJCAI*, Chambery, France, Morgan Kaufmann, pp. 1202–1207.