Using English for commonsense knowledge

Allan Ramsay
School of Computer Science
Univ of Manchester
Manchester M60 1QD, UK

Debora Field
Dept of Computer Science
Univ of Sheffield
Sheffield S1 4DP, UK

Abstract

The work reported here arises from an attempt to provide a body of simple information about diet and its effect on various common medical conditions. Expressing this knowledge in natural language has a number of advantages. It also raises a number of difficult issues. We will consider solutions, and partial solutions, to these issues below.

1 Commonse knowledge

Suppose you wanted to have a system that could provide advice about what you should and should not eat if you suffer from various common medical conditions. You might expect, at the very least, to be able to have dialogues like (1).

(1) a. User: I am allergic to eggs.
Computer: OK
User: Should I eat pancakes
Computer: No, because pancakes contain eggs, and eating things which contain eggs will make you ill if you are allergic to eggs.

b. User: My son is very fat.
Computer: OK
User: Should he go swimming.
Computer: Yes, because swimming is a form of exercise, and exercise is good for people who are overweight.

c. User: I have scurvy.
Computer: OK
User: Is eating berries good for me?
Computer: Yes, because berries contain vitamin C and eating fruit which contains vitamin C is good for people who scurvy.

Proceedings of the 8th International Conference on Computational Semantics, pages 181–194,
Tilburg, January 2009.

These are comparatively simple dialogues, requiring a very limited amount of knowledge about foods and medical conditions. As we will see, however, dealing with them does require a remarkable amount of knowledge about language.

The framework we are using makes a number of very basic assumptions about how you design a system to deal with such dialogues.

- To give appropriate answers to these questions you have to consider whether the available information supports *or contradicts* the queried proposition.
- In order to see whether the available information supports or contradicts a proposition you need a body of domain knowledge, and you need to be able to reason with it.
- Natural languages provide numerous ways of saying almost identical things. There is, for instance, almost no difference between *'I am allergic to eggs'* and *'I have an allergy to eggs'*. You therefore have to have a way of dealing with paraphrases.

We will explore each of these issues in turn below.

2 Answering questions

We will concentrate here on polar (YES/NO) questions. We also take the very simple view that when someone asks a polar question it is because they want to know whether the proposition encoded in the question is true or not. Someone who asks *'Is it safe for me to eggs?'* wants to be told *'Yes'* if it is and *'No'* if it is not. We have explored the nature of WH-questions elsewhere (Ramsay & Seville, 2001), and we have discussed situations where people use language in indirect ways (Ramsay & Field, 2008), but for the moment just trying to answer polar questions will pose enough problems to keep us occupied.

In order to answer such a question by saying *'Yes'* you have to see whether *'It is safe for the speaker to eat eggs'* follows from what you know about the speaker and your general knowledge. You cannot, however, say *'No'* simply because your attempted proof that it is safe failed. If you failed to prove that it is safe you should then see whether you can prove that it is not. If, and only if, you can then you should say *'no'*.

In general, then, answering a polar question may involve two attempted proofs–one aimed at showing that the proposition under discussion is true

and then possibly a second aimed at showing that it is false. If you are lucky you might discover evidence that the proposition is false while you are trying to show that it is true, but in general you may have to attempt two proofs.

In order to carry out proofs you need an inference engine. Inference engines come in all sorts of shapes and sizes–fast or slow, sound or unsound, complete or incomplete–and they can be applied to a variety of knowledge representation schemes. The choice of representation scheme, and of the inference engine that you apply to it, will depend on what you want to do with it. For our current task we assume that soundness is crucial, since you really would not want a medical advice system to give wrong advice; and that the representation scheme has to be highly expressive, since the relations involved are subtle and need to be represented very carefully.

This leads us to a very classical architecture: we construct formal paraphrases (logical forms, LFs) of the user's statements and questions, and we use a theorem prover to investigate the status of the propositions encoded in the user's questions. We are, in particular, not following the pattern matching path taken in most 'textual entailment' systems, since the informal rules used in such systems are not guaranteed to be sound.

We use 'property theory' (Turner, 1987; Chierchia & Turner, 1987) as our formal language. There are very strong grounds for believing that natural language is inherently intensional (we will see some examples below, but the simple presence of verbs of propositional attitude is hard to cope with unless you allow some degree of intensionality). There are a number of logics which allow for a degree of intensionality–the typed logic used by Montague (Montague, 1974; Dowty et al., 1981), the notion of non-well-founded sets (Aczel, 1988) used in situation semantics (Barwise & Perry, 1983), Bealer (1989)'s intensional logic, and so on. We choose property theory because one of Turner's axiomatisations draws a very strong analogy with modal logic which in turn suggests ways of adapting standard first-order theorem proving techniques for it. We have developed a theorem prover for a constructive version of property theory along these lines (Ramsay, 1995; Cryan & Ramsay, 1997), and have shown that it is sound (Ramsay, 2001). No theorem prover for a logic with this degree of expressive power can be complete–property theory, like default logic, is worse than first-order logic in this respect, in that it is not even recursively enumerable. Practical systems for first-order logic, however, do not proceed by enumerating all the theorems. They do their best, and if they cannot find an answer within a reasonable period of time then they give up. This the only sensible thing to do, and it is just as sensible when reasoning with more expressive languages.

We do not, however, just want to find out whether the answer to the user's question is *'Yes'* or *'No'*. We would also like to provide them with some explanation of how we arrived at our conclusion. It is much better to answer *'Should I eat pancakes?'* with *'No, because pancakes contain eggs, and eating things which contain eggs will make you ill if you are allergic to eggs.'* than just by saying *'No'*. The user will be more likely to accept the system's answer if it comes with some supporting explanation, and they may also be able to generalise from the explanation to cover other cases.

Where might we get such explanatory material from? The obvious place to look is in the trace of the proof that led to the conclusion. The proof tree contains the facts and rules that the system used in arriving at its conclusion. Showing these facts and rules to the user would let them see why the system believes that the queried proposition is true or false, and lets them judge the trustworthiness of what the system says.

There are two difficult problems to be addressed here. The first is that the proof tree will contain a mixture of things that the user themselves said, things are blindingly obvious and things that the system suspects that the user might not know. The explanation should probably concentrate on things that the user might not have been aware of, so we should be looking for items in the proof tree that the system believes the user may not know. In other words, we need an epistemic version of property theory, and we need to be able to inspect facts and rules to see who has access to them. We will not discuss this further here, except to note that in the concrete examples below we are *not* doing this, so that the support for the system's conclusions currently includes material that the user would actually already be aware of.

The second problem is that it is extremely difficult to generate natural language text from arbitrary logical expressions. We use standard compositional techniques to build our logical forms on the basis of the form of the input text (van Genabith & Crouch, 1997; Konrad et al., 1996). However, as with virtually any practical theorem prover, we then perform a number of transformations (Skolemisation, distribution of negation, splitting rules with conjunctive heads) to our logical forms in order to make them amenable to the theorem prover. By the time we have done this there is very little hope of using the compositional semantics in reverse to generate natural language text from elements of the proof tree. There is, in particular, no chance of using head-driven generation (Shieber et al., 1990) to produce text from elements of the logical form, since this approach requires that the elements of the logical form be unifiable with the meanings of lexical items in order to drive the selection and combination of words. This is just not feasible with

the elements of a proof tree.

Where do the facts and rules that appear in the proof tree come from? We clearly have to specify them in advance in some form. Some of this information comes from the user, in the form of statements about their conditions, but most of it will have to provided explicitly.

We can do this in a variety of ways. We could try to use some existing resource–WordNet, CYC, some medical ontology. It turns out that these resources, or at least the publicly available ones, lack a great deal of what we need. Very few such resources contain the kind of rules you need for answering questions such as the ones in (1). Lexical resources contain information about relations between words. WordNet, for instance, provides hypersensitivity reaction, hypersensitivity, sensitivity, susceptibility, condition, state, attribute, abstraction, entity as hypernyms of *'allergy'*–all perfectly sensible hypernyms, but not all that useful for answering (1a). Likewise the only mention of allergy or allergic in the ontology in OpenGalen (version 7, downloaded 09/10/08) says that an allergy is a kind of pathology, and SnoMed has 'propensity to adverse reactions' and disease as hypernyms, and a variety of links to special types of allergies and other related conditions.

This is not, of course, an exhaustive search of all potentially relevant ontologies, but it does suggest that the kind of information stored in a typical ontology is not what we require for answering our questions. It is, however, extremely interesting to note that WordNet contains an English gloss for *'allergy'* as *'hypersensitivity reaction to a particular allergen; symptoms can vary greatly in intensity'*, OpenGalen contains the text *'Hypersensitivity caused by exposure to a particular antigen (allergen) resulting in a marked increase in reactivity to that antigen upon subsequent exposure sometimes resulting in harmful immunologic consequences.'* and SnoMed provides very brief English glosses. It seems as though when ontology designers want to say what a term really means, they resort to natural language.

It also seems as though this kind of ontology fails to include 'commonsense' knowledge, e.g. that if you are allergic to a foodstuff then you should not eat it. We need this kind of information in order to answer questions. The prevalence of natural language glosses in ontological resources of this kind, suggests that expressing it in natural language might be a good idea.

Using natural language to express the knowledge that we need has a number of advantages:

- It is comparatively easy. Most people (even logicians and semanticists!) generally find it easiest to express themselves in their native language. It is much easier to write *'If you are allergic to something*

then eating it will make you ill' than to express the same rule in some formal language.

- Linking knowledge that has been expressed in natural language with facts and queries that have been expressed in natural language obviates the need for a set of terminological mappings between domain knowledge and natural language. The vocabularies used in terminological databases tend to have a superfical resemblance to words in natural languages, but the mapping is seldom exact, and indeed the types associated with such terms are often quite different. If all your knowledge is expressed in natural language then this kind of problem can be avoided.

- Finally, it makes it much easier to generate answers. If we keep a link between surface text and logic we can retrieve the surface text from the proof tree. This does not entirely solve the problem of producing coherent natural language answers, but it does make it much simpler.

3 English with variables

We therefore want to try writing down various commonsense rules in English. To make it slightly easier to write rules, we allow variables in various places. Thus we write (2b) rather than (2a).

(2) a. Eating fruit which contains vitamin C is good for you if you have scurvy

b. Eating fruit which contains vitamin C is good for X if X has scurvy

This is helpful here simply to get around the fact that *'you'* is normally taken to be a reference to the hearer, whereas in (2a) it is being used in a rather generic way. Rather than allowing *'you'* to be ambiguous in this way, we simply allow variables to be used in natural language.

The logical form we obtain for (2b) is shown in Fig. 1. There are a number of things to note about Fig. 1:

- It's enormous. Reading it you can see how it relates to (2b) itself, but producing something like this by hand would certainly be a major challenge. However, if we have a set of rules that explain the relationship between structural (lexical and syntactic) choices and semantics then we can obtain Fig. 1 directly from the parse tree of (2b). *This and*

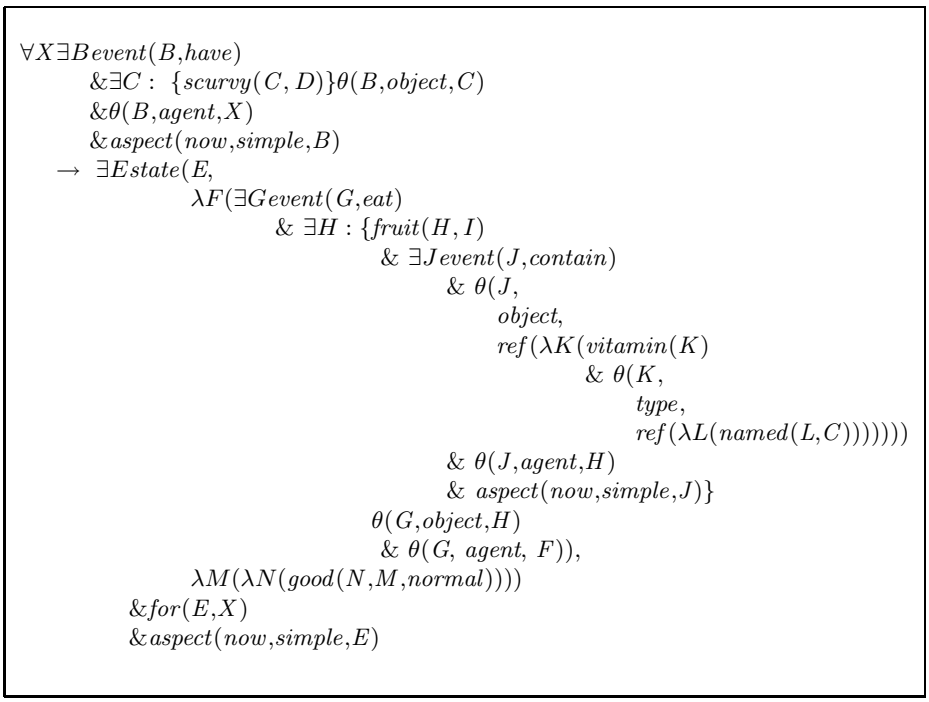

Figure 1: Logical form for (2b)

all other logical forms in this paper were produced by applying compositional rules to the first parse we obtained from the relevant texts. So although it is indeed enormous, and complicated, all we have to do is write the rule in English and the system will take care of the rest.

- The analysis of *'eating fruit which contains vitamin C is good for you'* introduces a relationship between two intensional objects, namely 'the kind of event where someone eats fruit which contains vitamin C' and 'the property of being good'. This has significant consequences for the kind of inference that is required. We will explore this further below.

Once we allow variables in places where you might expect an NP, it becomes tempting to introduce them in other places. The consequences of doing this are interesting:

(3) a. Eating something will make you ill if you are allergic to it

b. Eating P will make X ill if X is allergic to P

Again the second version of the rule sidesteps some tricky details to do with pronouns, but the formal paraphrase throws up some awkward issues.

$$
\begin{array}{l}
\forall X \forall P \exists C\, state(C,X,\lambda D(\lambda E(allergic(E,D,normal)))) \;\&to(C,P) \\
\qquad \&aspect(now,simple,C) \\
\quad \rightarrow\ \exists F : \{future(now,F)\} \\
\qquad\quad \exists B\, event(B,make) \\
\qquad\qquad \&\theta(B,object,X) \\
\qquad\qquad \&\theta(B,object1,\lambda G(\lambda H(ill(H,G,normal)))) \\
\qquad\qquad \&\theta(B, \\
\qquad\qquad\quad agent, \\
\qquad\qquad\quad \lambda I\ \exists J\, event(J,eat) \\
\qquad\qquad\qquad \&\exists K :\ \{P : K\}\theta(J,object,K) \\
\qquad\qquad\qquad \&\theta(J,agent,I)) \\
\qquad\quad \&aspect(F,simple,B)
\end{array}
$$

Figure 2: Logical form for (3b)

The new problem in Fig. 2 is the paraphrase of *'eating P will make X ill'*, where *'P'* stands for something like *'something of type P'*. In other words, P here is a variable noun rather than a variable NP.

Under almost any analysis, nouns denote kinds rather than individuals. But that means that (3b) involves quantification over kinds, which is again very intensional.

4 Inference

Constructing LFs which involve relations between intensional entities is not problematic. As noted above, we know there are formal languages which allow for intensionality, so all we have to do is choose one of these for our LFs. Indeed, most approaches to compositionality exploit λ-abstraction and β-reduction, so the intermediate objects that are constructed are inherently intensional anyway. Anyone who takes the interpretation of *'man in a park'* to be something like $\lambda A(\exists B :\ \{park(B,C)\}\exists D(man(D,E)\ \&\ in(D,B))\&(A : D))$ (i.e. the standard Montague representation) is using an intensional language as an intermediate representation.

Problems only arise when we try to reason with representations of this kind. There is, however, very little point in making LFs if you are not going to reason with them, so we do have to worry about it. To see a concrete example, reconsider (1c), repeated as (4):

(4) [User: I have scurvy.
Computer: OK
User: Is eating berries good for me?
Computer: Yes, because berries contain vitamin C and eating fruit which contains vitamin C is good for people who scurvy.

Our rule about scurvy says that events of a certain kind are good for people who have scurvy. The description of these events occupies an *argument* position in the LF, as one of the terms describing the general state of affairs that holds if someone has scurvy.

In Prolog-based first-order theorem provers, you determine whether you can use a rule to prove a goal by *unifying* the arguments of the goal with the arguments of the consequent of the rule[1]. In the current case, this would mean unifying the terms describing 'eating fruit which contains vitamin C'-events and 'eating berries'-events.

Clearly these descriptions will not unify. What we have to do is to accept that the rule can be used with terms that describe subsets of the classes that appear in argument positions.

We do not want to do this everywhere. This is a characteristic of the rule about the link between vitamin C and scurvy, not a general characteristic of all rules. When we want to allow for this, we have to say so explicitly.

We therefore include a rule which says that the idea that events of some kind are good or bad or safe or ... for you is 'downward entailing': if eating fruit which contains vitamin C is good for you then eating berries is good for you, because the set of 'eating berries'-events is a subset of the set of 'eating fruit which contains vitamin C'-events. This rule is given in Fig. 3.

$$\forall B \forall C \forall D : \{\forall F : \{state(B,F,E,for(C)) \& (\forall G(D{:}G) \rightarrow (F{:}G))\} \; state(B,D,E,for(C))$$

Figure 3: Downward entailment for states

Fig. 3 says that if events that satisfy the description F satisfy the property E for the individual C, then so do all events whose description G entails F.

We need a similar rule to say that this kind of relationship is upward entailing in the third argument–that anything which is good for you, for

[1]Prolog-based theorem proving is a special case of resolution theorem proving. In general resolution theorem provers you have to unify some positive literal in one clause with a negative one in another. For simplicity we will talk in terms of goals and consequents, but the analysis would apply to other resolution-based engines.

instance, is also safe for you. Rules like these exploit a notion of 'guarded intensionality', in that they are only applicable when you already know what properties you are interested in. They thus essentially act as schemas for first-order rules. If we only use them backwards, in situations where we know what properties we are interested in, they can be applied in a fairly controlled way, and hence do not introduce wild combinatorial problems.

This is not the only kind of intensional rule that we need, but it does cover a substantial number of interesting cases. The theorem prover described in (Ramsay, 2001) can cope with more general intensional rules, but guarded rules of this kind can be dealt with more efficiently than general rules, and they are particularly useful for axiomatising the phenomena that interest us.

5 Paraphrases and other lexical relations

It is clear that we need to treat with a variety of relations between everyday terms. We will return, as an example, to (1b), repeated here as (5).

(5) User: My son is very fat.
Computer: OK
User: Should he go swimming.
Computer: Yes, because swimming is a form of exercise, and exercise is good for people who are overweight.

The computer's answer to the user's question clearly depends on an understanding that if something is good for you then you should do it. To say this is not, of course, to provide a complete characterisation of the meaning of *'should'*. It is just a piece of commonsense. Nonetheless, for a system to be able to cope with (1b) it has to have access to this piece of commonsense. Fig. 4 shows the axiomatisation of this notion: if events of the kind described by B are good C, then if I describes an action whose performance entailed that B held for C then I should happen.

$$\forall B \forall C \forall D : \{state(D,B,\lambda E(\lambda F(good(E,F,G))),for(C)) \\ \&(\exists H(I{:}H)) \rightarrow (B{:}C))\} \\ \exists J : \{aspect(now,simple,J)\} should(J,I)$$

Figure 4: If something's good for you then you should do it

There are a variety of other very basic elements of commonsense which have to be captured, and which are not generally included in formal ontologies. We need to know, for instance, that something cannot be both good and bad, and that dangerous things are bad, and so on. Some of these can only be axiomatised manually, but the aim is to keep things that have to be encoded manually to a minimum. As noted earlier, writing axioms in English is generally easier and it also makes them easily available for use in explanations. Very basic things like the fact that things cannot be both good and bad are unlikely to be required for explanations, even if they do take part in proofs, so the fact that they are unavailable for this purpose does not matter.

Some of these basic relations turn out to be bi-equivalences, or as near to bi-equivalences as makes no difference. It is extremely difficult, for instance, to articulate any difference between (6a) and (6b).

(6) a. I have an allergy to eggs.

b. I am allergic to eggs.

We could take account of this by introducing a pair of implications: *'X has an allergy to P if X is allergic to P'* and *'X is allergic to P if X has an allergy to P'*. This would work, in the sense that we would be able to use these two constructions interchangeably, but it would slow the inference engine down considerably. The presence of any pair of rules of the form $P \rightarrow Q$ and $Q \rightarrow P$ will inevitably slow any theorem prover down, since any attempt to prove Q is likely to lead to an attempt to prove P, which will in turn lead to an attempt to prove Q. It is not difficult to catch simple loops of this kind, but it is better to avoid them in the first place if possible.

We therefore use rules like this as part of the normal-forming process. Construction of normal forms generally involves application of bi-equivalences where one side has a form which is particularly well-suited to the needs of a particular theorem proving algorithm. In resolution, for instance, the rules $\neg(P\&Q) \leftrightarrow (\neg P \vee \neg Q)$ and $\neg(P \vee Q) \leftrightarrow (\neg P\&\neg Q)$ are used during the construction of the normal form because resolution looks for matching positive and negative literals, so axioms that can be used to ensure that the only negation signs appear at the lowest possible level are useful.

The point of normal-forming, then, is to ensure that bi-equivalences are applied just once, and in just one direction. We thus apply bi-equivalences like the one between (6a) and (6b) during the construction the construction of logical forms. This lets us cope with the fact that natural languages

typically provide a range of ways of saying virtually the same thing without incurring the expense of applying rules which potentially lead to loops when we are carrying out inferences.

There is a complication here. The system needs to realise that (7a) and (7b) are also the same (and likewise for other variations).

(7) a. I have a severe allergy to eggs.

b. I am severely allergic to eggs.

Dealing with this requires considerable care in the design of logical forms. Space precludes a deeper discussion of this issue, but this is something we have to take care over.

6 Conclusions

The work described here covers very similar ground to work in textual entailment (Dagan et al., 2005), in that we want to draw inferences based on facts and rules expressed in natural language. Producing logical forms and then using a theorem prover to carry out the required inference leads to more reliable conclusions, since we can check that the theorem prover is sound, and hence we can rely on the conclusions that it draws. It also leads to deeper chains of inference, since the pattern matching algorithms generally employed for textual entailment do not lend themselves to repeated application.

The approach outlined here does involve a number of risks. We might not be able to express all the knowledge we want in natural language; we might not be able to produce logical forms from our natural language rules; when we have more rules the theorem prover might not be able to cope.

The last of these is the most dangerous. If there are rules which we cannot express in natural language, or where we cannot convert the natural language into a logical form, we can always express them directly in property theory (or property theory with procedural attachment of appropriate, e.g. mathematical, rules (Steels, 1979)). How long will it take when there are large numbers of rules? The proofs for the examples here take around 0.1 sec. Most of this time is spent investigating intensional rules. Most of the commonsense knowledge, however, is represented as Horn clauses. Indeed it is represented as pure Prolog. The speed of Prolog programs is not affected by the number of clauses that are present, so we are confident that adding more rules will have very little effect on the performance *so long as they can be represented as Horn clauses.* The key issue, then, is how many new

intensional rules we will need. Only time will tell, but we are hopeful that we will retain a reasonable level of performance even when we have a more substantial set of rules. If not, we will just have to make the theorem prover faster.

References

P. Aczel (1988). *Non-Well-Founded-Sets*. CSLI Publications, Stanford.

J. Barwise & J. Perry (1983). *Situations and Attitudes*. Bradford Books, Cambridge, MA.

G. Bealer (1989). 'Fine-grained type-free intensionality'. In G. Chierchia, B. H. Partee, & R. Turner (eds.), *Properties, types and meaning: vol I, foundational issues*. Kluwer Academic Publishers, Dordrecht/Boston/London.

G. Chierchia & R. Turner (1987). 'Semantics and Property Theory'. *Linguistics and Philosophy* **11(3)**.

M. Cryan & A. M. Ramsay (1997). 'A Normal Form for Property Theory'. In *Proceedings of the 14th International Conference on Automated Deduction (CADE-14)*, vol. 1249 of *Lecture Notes in Artificial Intelligence*, pp. 237–251, Berlin. Springer-Verlag.

I. Dagan, et al. (2005). 'The PASCAL Recognising Textual Entailment Challenge'. In *Proceedings of Pascal Challenge Workshop on Recognizing Textual Entailment*.

D. R. Dowty, et al. (1981). *Introduction to Montague Semantics*. D. Reidel, Dordrecht.

K. Konrad, et al. (1996). 'An education and research tool for computational semantics'. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, pp. 1098–1102, Copenhagen.

R. Montague (1974). 'The proper treatment of quantification in ordinary English'. In R. Thomason (ed.), *Formal Philosophy: Selected Papers of Richard Montague*, New Haven. Yale University Press.

A. M. Ramsay (1995). 'A Theorem Prover for an Intensional Logic'. *Journal of Automated Reasoning* **14**:237–255.

A. M. Ramsay (2001). 'Theorem proving for untyped constructive λ-calculus: implementation and application'. *Logic Journal of the Interest Group in Pure and Applied Logics* **9**(1):89–106.

A. M. Ramsay & D. G. Field (2008). 'Speech acts, epistemic planning and Grice's maxims'. *Logic and Computation* **18**:431–457.

A. M. Ramsay & H. L. Seville (2001). 'Relevant Answers to WH-questions'. In *3rd International Conference on Inference in Computational Semantics*, pp. 73–86, Siena.

S. M. Shieber, et al. (1990). 'Semantic-Head-Driven Generation'. *Computational Linguistics* **16(1)**:30–42.

L. Steels (1979). 'Procedural attachment'. Tech. rep., MIT.

R. Turner (1987). 'A Theory of Properties'. *Journal of Symbolic Logic* **52(2)**:455–472.

J. van Genabith & R. Crouch (1997). 'How to glue a donkey to an f-structure'. In H. C. Bunt, L. Kievit, R. Muskens, & M. Verlinden (eds.), *2nd International Workshop on Computational Semantics*, pp. 52–65, University of Tilburg.

Appendix: commonsense rules

(8) a. eating P will make X ill if X is allergic to P.

b. exercise is good for X if X is overweight.

c. swimming is good for X if exercise is good for X.

d. walking is good for X if exercise is good for X.

e. eating fruit which contains vitamin C is good for X if X has scurvy.

f. X eats P if X eats something which contains P.

g. X is dangerous for Y if X will make Y ill.