

# Multilingual Discourse Processing

Bill Black, Andrew Conroy, Adam Funk,  
Allan Ramsay, Mark Stairmand, Paul Thompson

Department of Computation  
UMIST  
Manchester M60 1QD, UK  
dumas-tech@co.umist.ac.uk

## Abstract

The goal of the work reported here is to provide a system for handling spoken dialogues which can be easily ported to new languages and new application areas. This work is being carried within the Dumas project, which aims to explore a number of approaches to this task. The particular approach described here involves constructing fairly traditional ‘logical forms’ on the basis of structural analyses of the input speech and then using a mixture of meaning postulates, which relate terms in the language to underlying concepts, and general domain knowledge in order to interpret and respond to what has been said.

## 1 Introduction

We want to provide a generic framework within which we can develop spoken language dialogue systems which can be easily reconfigured to cover new languages or new application areas. The motivation for this is obvious: it’s hard work developing such a system, and if you can reuse any of what you’ve done for one language or application when you want to move to another then you will save some of that effort. How you go about doing it is rather less obvious.

Most existing practical spoken language systems work by developing fine-grained local grammars to cover the most likely utterances at each point in a dialogue, and associated (linguistic and extra-linguistic) actions with each such utterance. Toolkits such as the CSLU toolkit (McTear, 1998) and Nuance [www.nuance.com](http://www.nuance.com), to name just two, provide facilities for constructing such systems, and there is no doubt that this is a very effective way of producing a useable system fairly quickly.

This approach does, however, have a number of disadvantages:

- you have to write a grammar. There are good reasons for writing a number of local grammars, and switching between them depending on where in the dialogue you are (namely that doing so reduces the perplexity of the grammar, and hence increases the accuracy of the speech recognition element of the system). Writing grammars is, nonetheless, a challenging and time-consuming activity, and the more you can re-use existing grammars the better. This tends not to happen if the current system has been constructed by hand-crafting an application specific grammar and attaching actions directly to the rules, since you have to disentangle the rules from the grammar when you want to develop a new application.
- when the application becomes reasonably complex, it may be necessary to carry out substantial amounts of reasoning, firstly in order to make sense of what the user has said (e.g. for lexical or structural disambiguation) and then in order to work out how to respond. This can be awkward if the actions are attached directly to the grammar rules.
- in multilingual applications, a good deal of this reasoning is unchanged when you switch languages. Again, if the actions are attached directly to the rules it can be difficult to share the same inference processes across languages.

The work reported here attempts to decouple the structural analysis of the speech signal (i.e.

word recognition and morpho-syntactic analysis) from the interpretation and reasoning. If we can do this, then we can reuse the the same grammar for a number of different applications, and we can reuse the same application engine for a number of different languages. The risk, of course, is that it may prove to be so difficult to construct a language-independent application-independent interpretation that any potential savings vanish into thin air.

## 2 Architecture

The Dumas project employs a generic agent-based architecture in order to make it easy to connect self-contained components (Turunen and Hakulinen, 2000). Within the project as a whole a number of different configurations of agents have been proposed. For the work reported here we have used the following agents, with control being passed more or less sequentially between them:

**Speech recognition:** speech recognition is carried out using a standard black box speech recogniser, with a grammar written to cover typical utterances for the domain. We currently use Nuance, with grammars written to cover utterances recorded during a series of Wizard-of-Oz experiments. The grammars are domain dependent, but the actions associated with the rules do nothing but output the recognised tokens to the next agent. It is thus easy to reuse those parts of the grammar that are common across different domains (e.g. if you were writing a system to provide technical support, it is likely that a number of standard phrase types will occur even if the domain-specific content words change). Other speech recognition engines could be employed at this point in the system (if, for instance, Nuance did not cover a specific language), and so long as they can be configured to produce the token sequence as output (which is the easiest thing to make them do) then they can be plugged in at this point instead.

**Morpho-syntactic analysis:** the structural properties of the utterance encode a great deal of its meaning. We therefore need an accurate analysis of these. Again we use

an off-the-shelf system, namely the Connexor parser (Tapanianen and Jarvinen, 1997), and again this could be replaced if we needed to cover a language which is not provided by the Connexor parser (in fact we did start out with a different parser, but the one we were using did not cover Swedish or Finnish).

If we want to be able to switch different parsers in and out and yet retain the ability to construct ‘logical forms’ compositionally, we need to be able to translate between different syntactic formalisms. The construction of logical forms is based on the approach outlined in (Ramsay and Seville, 1999), where we use the  $\lambda$ -calculus to describe the meanings of individual lexical items and we include ‘functional glue’ to specify what should be done with a given lexical item when it is used for a particular purpose (so that we use different glue when using a noun as a modifier in a noun-noun compound rather than as the head of an NP). This is more restricted than the notion of glue proposed by (Dalrymple et al., 1996) and extended by (Crouch et al., 2001), but it does provide a route by which we can accept syntactic analyses in a range of different formats and still construct the same logical forms.

**Semantic interpretation:** on the basis of the morpho-syntactic analysis we construct ‘unanchored logical forms’ (ULFs), which capture as much of the literal meaning of the utterance as possible. These ULFs contain terms corresponding to definite NPs. These ‘referring terms’ have to be anchored in the context, either by finding an item that satisfies the relevant properties, or by proving that there is one without explicitly finding it in the context, or by ‘accommodating’ it. How this is done, and what the system does when a reference is accommodated, is discussed further below.

**Discourse tree:** each utterance, by either the user or the system, is used to extend a growing discourse tree. Each node of the tree is a partial model, containing facts that have been inferred on the basis of the current utterance, the information available

in the previous nodes of the tree, and the background (domain) knowledge available to the system. Numerous authors have suggested using a structure of this kind for representing discourses, particularly in order to control the search for pronominal referents e.g. (Grosz et al., 1995; Hitzeman and Poesio, 1998). A wide range of cues for deciding how to attach new nodes to the tree. We currently just start a new branch of the tree each time the user asks a question. We have investigated a number of more sophisticated sets of cues (Seville, 1999), but for present purposes this very simple approach seems to be adequate.

**Response planning:** the discourse tree contains everything we can determine about the user's utterance: the simple literal meaning, the surface linguistic act, and anything that follows from these together with the systems domain knowledge. From this we have to work out what to do – if we can't make a plan on the basis of this information then we can't make one at all, since there is nothing else. We start by specifying a set of goals, which arise directly from what we understand about the utterance. We add these to a goal stack, and then invoke a straightforward STRIPS-style planner to choose a sequence of actions which will achieve them. These actions may be linguistic or non-linguistic. Our testbed application allows the user to interact with his or her mailbox: in this context the system may be required to carry out linguistic actions (e.g. reading a message or part of a message) or non-linguistic actions (e.g. deleting a message) or perhaps both.

The response planner produces a sequence of actions to be performed by appropriate agents. There is no reason to assume, however, that every action proposed by the planner will actually be carried out. Some actions may turn out to be impossible (such as deleting a message that you've already deleted), others may simply not happen because the user interrupts what is going on and starts a new dialogue or sub-dialogue. It is therefore essential that each agent updates the discourse model when it has performed one of the required actions, so that

the system continues to have a model of the world that matches both reality and the user's model.

### 3 Logical Form

We assume that a logical form is a formula of some suitable logic which captures as completely and accurately as possible the information encoded in the form of the utterance (i.e. in the choice and arrangement of words). The logical form is essentially an abstraction away from the details of the surface form, capturing the information that can be used as the basis for action and throwing away the details of how this information was realised by the surface form. Any information that is lost at this point is lost forever, and it is therefore crucial that the morpho-syntactic analysis preserves all the semantically relevant features of the text.

The logical form, then, is constructed by inspecting the output of the morpho-syntactic analysis. Since we want to be able to switch grammars and parsers, we have to have some way of coping with the fact that different syntactic analysers use different formalisms and different annotations. The vast majority of such formalisms, however, produce structures which can be converted to labelled dependency trees, where the dependency arcs are assigned functional labels drawn from finite set of labels. The two parsers that we have worked with most both produce such trees as their output anyway, but the output of any parser which produces a phrase structure tree can be converted to a dependency tree by taking one element of any sub-tree to be the head of that tree.

Labelled dependency tree analyses can be used as the basis of a compositional approach to semantics as follows:

- each lexical item is assigned a meaning, which will typically be a  $\lambda$ -abstraction. This follows standard practice in post-Montague approaches to semantics, where a noun like '*man*' might be given  $\lambda(X, man(X))$  as its meaning and an adjective like '*new*' might mean  $\lambda(P, \lambda(X, (P : X) \& new(X)))$ . Most of these items are closely related to Montague's original suggestions, at least for simple cases like simple NPs and straightforward clauses.

- each item is assigned a ‘glue’ for each functional label. So an adjective might be assigned  $\lambda(U, \lambda(V, V : U))$  as the glue to be used when it is functioning as a modifier, and  $\lambda(P, \lambda(X, P : X))$  as the glue to be used when it is functioning as an argument. The glue associated with a functional label replaces the rule-to-rule mapping of the original Montague treatment, which is not available in the kind of highly lexical grammars in widespread common use.

This approach minimises the difficulties that arise when you switch grammars or languages. In order to construct logical forms in a new framework all we have to do is assign glue to each functional label and each configuration of non-functional labels. As an example, consider the following sentence::

(1) Do I have any messages from Adam

The output from the Connexor parser for this sentence looks roughly as follows (some detail has been omitted to save space).

	lemma	function	description
1	do	v-ch:3	@+FAUXV AUX V PRES
2	i	subj:1	@SUBJ PRON SG1
3	have	main:0	@-FMAINV VA V INF
4	any	det:5	@DN N DET
5	message	obj:3	@OBJ NH N NOM PL
6	from	mod:5	NOM N PREP
7	adam	pcomp:6	P NH N NOM SG +ind

The column headed ‘function’ specifies the dependency relation, so that `subj:1` in row 3 says that ‘*I*’ is the subject of ‘*have*’. We specify that the glue for the `subj` relation is  $\lambda(E, \lambda(S, \lambda(Y, \theta(Y, \text{subj}, S)) \& (E : Y))))$ , and then this glue will bind the meaning of ‘*have*’ (which denotes an event of type *have*, specified as  $\lambda(E, \text{have}(E))$ ) and ‘*I*’, which, following standard practice, denotes the property of being true of the speaker –  $\lambda(P, P : \text{ref}(\lambda(X, \text{speaker}(X))))$ .

Given an appropriate set of interpretations of functional and non-functional labels, we arrive at

$$\forall A : \{en\_message(A) \& en\_from(A, \text{ref}(\lambda B(\text{named}(B, 'Adam'))))\}$$

$$\exists D : \{\text{aspect}(\text{simple}, \text{ref}(\lambda E(E \in \text{interval} \& \text{overlap}(\text{ref}(\lambda F(\text{now}(F))), E))), D)\}$$

$$\theta(D, \text{subj}, \text{ref}(\lambda G(\text{speaker}(G))))$$

$$\& \text{event}(D, en\_have)$$

$$\& \theta(D, \text{obj}, A)$$

$$\rightarrow \text{answer}(yes)$$

$$\& \text{utt}(\text{query})$$

as the interpretation of (1). This says that the utterance was a query, and that if there is something which is a message from the person called Adam and which I have then the answer is ‘*yes*’.

The details of this logical form do not concern us here – other people might propose, for instance, different treatments of tense and aspect. The crucial points are as follows:

- The surface mood has been encoded in the logical form. If we do not record the fact that this is a polar question which should be answered affirmatively if I have a message from Adam here, we will have to encode it somewhere else and then re-link the propositional content to the mood at a later date. Since we have derived both the mood and the propositional content from aspects of the dependency tree and we are going to need to consider them together when we come to plan our response, it seems sensible to capture them both in the same logical form.
- The logical form contains terms corresponding to the open class words in the sentence with the prefix ‘*en-*’ added. These terms will be linked to a set of underlying concepts which are largely language independent. For each language we have a set of meaning postulates which link the language dependent terms to the same set of underlying concepts.

By deriving the logical form by assigning glue to functional and non-functional labels, we can (fairly) easily adapt to a new language. Consider (2):

(2) har jag några meddelanden från Adam

The Connexor analysis for this is as shown below:

	lemma	function	description
1	ha	main:0	@MAIN V ACT IND
2	jag	subj:1	@NH PRON UTR
3	några	attr:4	@PREMOD PRON
4	meddelande	obj:1	@NH N NEU PL
5	från	advl:1	@PREMARK PREP
6	adam	pcomp:5	@NH N UTR SG

It is immediately obvious that this analysis assigns basically the same dependency tree to this sentence as the tree that was assigned to (1). It is almost equally obvious that the fine detail of the two trees is different – that ‘*any*’ is labelled as a determiner playing the functional role of determiner with respect to the noun ‘*message*’, whereas according to the output of the Connexor parser ‘*några*’ is seen in this context as a premodifying pronoun (@PREMOD PRON) playing the rather more generic role of **attr** with respect to its noun. Thus even if we use one family of systems for our syntactic analysis we may need to change the fine detail of the mapping between labels and glue. Nonetheless, this is easier than writing a complete new semantic analyser for each new language.

The logical form that we obtain for (2) is shown below:

$$\begin{aligned} \exists A : \{ & se\_meddelande(A) \\ & \& se\_frn(D, ref(\lambda G(named(G, 'Adam')))) \} \\ \exists B : \{ & B \in interval \\ & \& overlap(ref(\lambda C(now(C))), B) \} \\ \exists D : \{ & aspect(simple, B, D) \} \\ & event(D, se\_ha) \\ & \& \theta(D, subj, \lambda(E, speaker(E))) \\ & \& \theta(D, obj, A) \\ \rightarrow & answer(yes) \\ & \& cutt(query) \end{aligned}$$

This logical form contains terms prefixed with ‘*se\_*’ to mark the fact that they are derived from Swedish surface forms, but apart from that it has a lot in common with the form for (1). The major difference is that the Swedish determiner ‘*några*’ is a simple indefinite, which introduces an existential quantifier into the logical form, whereas the English determiner ‘*any*’ is a rather strange universal quantifier. But note that the logical form (1) has the general form  $\forall X(P : X \rightarrow answer(yes))$ , where the universal quantifier has wide-scope over the mood, whereas the form for (2) is of the form  $(\exists X(P : X)) \rightarrow answer(yes)$ . These two are logically equivalent – they both say that you should answer ‘*yes*’ if there is something that

satisfies *P*. Thus by using general properties of dependency trees and open class words but paying close attention to the fine detail of the closed class words we can obtain appropriate logical forms for sentences from different languages or from dependency trees obtained by different parsers and grammars.

Of course it is not quite as simple as this: unless the grammar writers are careful to use the same labels for the same things when developing grammars for different languages, you may find that you have to write parallel sets of label→glue rules, and this is even more significant when you switch grammatical frameworks. Nonetheless it is certainly easier to write new sets of rules to exploit an existing view of how to stick the meanings of different open class words together than to develop an entirely new approach to semantics for each language or framework. Proceeding this way also guarantees that all the logical forms have the same basic structure and the same set of underlying ontological assumptions, so that we do not have to redesign the reasoning engine when we switch languages or syntactic frameworks.

## 4 Discourse Tree

The logical forms described in section 3 are rather bare. They contain unanchored referring terms, and they contain terms that have not been directly connected to domain concepts. We take the view that the user’s utterance should be interpreted in the light of all the information available at the time when it is uttered, so that each time anyone says something then the hearer should update their model of the current state of the discourse. We do this by using background knowledge to flesh out the information contained in the logical form, partly by anchoring the referring expressions appropriately and partly by taking account of additional propositions that follow from our background knowledge.

The background knowledge combines ‘meaning postulates’, which link lexically derived terms to domain concepts, and general domain knowledge. Meaning postulates are generally fairly straightforward, e.g. the following pair link the English word ‘*message*’ and the Swedish word ‘*meddelande*’ the domain concept *message*.

$$\forall A(\text{message}(A) \leftrightarrow \text{en\_message}(A))$$

$$\forall A(\text{message}(A) \leftrightarrow \text{se\_meddelande}(A))$$

Meaning postulates are not always as straightforward as this. There may be extra contextual information on either side of the equivalence – a surface form might be ambiguous, so that the left-hand side might contain a disjunction, it might only correspond to the underlying concept in certain linguistic contexts, in which case the right-hand side might contain extra side conditions, ... The general approach of providing axiomatic links between words and concepts works straightforwardly when the link is straightforward, but it leaves room for considerable subtlety in complex cases (we can, for instance, follow (Cruse, 1986)'s suggestion that words depict semantic traits by using default rules in these meaning postulates).

Domain knowledge tends to be more complex. We might, for instance, note that if you have a message from someone then they must have sent it to you: if  $B$  is a message from  $C$  to  $A$  then there must be a sending event  $D$  of involving  $B$  as the object,  $C$  as the agent and  $A$  as the recipient:

$$\forall B : \{\text{message}(B)\}$$

$$\forall C : \{\text{from}(B, C)\}$$

$$\forall A : \{\text{to}(B, A)\}$$

$$\exists D \text{event}(D, \text{send})$$

$$\quad \& \theta(D, \text{subj}, C)$$

$$\quad \& \theta(D, \text{obj}, B)$$

$$\quad \& \theta(D, \text{dat}, A)$$

Encoding domain knowledge in this form makes it possible to reason about the consequence of utterances, and hence to go beyond what was actually said. This gives us extra flexibility, for instance in recognising that ‘*Do I have any new messages from Adam?*’ and ‘*Has Adam sent any new messages?*’ mean the same thing.

On the basis of this information, we build a local discourse model containing the consequences of the anchored logical form. The local discourse model contains a set of propositions, which are basically the components of the logical form with the referring expressions instantiated and anything that follows in the current

context from the logical form by the application of default rules (we do not store things that can be inferred on the basis of standard rules, since in practice it seems to be more efficient to infer such things when we need them than to do undirected forward reasoning and cache the results). The local discourse model also contains information about the discourse status of the individual items mentioned in the current sentence – were they introduced in it, or referred to by definite NPs (and if so what kind of NPs), were they introduced as a result of accommodation, ...? In particular, items corresponding to problematic definite NPs are flagged – was there no appropriate referent, or were their multiple potential referents? Such cases reveal misunderstandings between the user and the system, and require responses aimed at sorting out where the conversation has gone wrong rather than actual domain actions:

- (3) a. Do I have any messages from Adam?  
 b. You have one new message from Adam.  
 c. Read the first.  
 d. First message from Adam: I’ve just sent you those logical forms. ...  
 e. Read the next.  
 f. I’m sorry, there is nothing that fits the description ‘*the next*’
- (4) a. Has Bill sent me any messages?  
 b. You have three new messages from Bill?  
 c. Read the message.  
 d. There are several things that fit the description ‘*the message*’. Which one do you want me to read?

Constructing appropriate responses is not a trivial task.

## 5 Conclusions

The work outlined above provides the basis for developing spoken language dialogue systems which can be (fairly) easily adapted to work with new languages or in new application domains. It is unlikely ever to be a trivial matter, but separating out the stages that get you from the structural analysis of the input speech signal to the pure unanchored logical form, and from this to an interpretation of the utterance in

context makes it possible to replace these components individually. In particular, assuming labelled dependency trees as a common framework for syntactic description helps preserve independence from the details of any specific morpho-syntactic analyser, since the vast majority of contemporary syntactic frameworks are isomorphic to this framework anyway: ‘all’ you have to do is translate the labels into some lowest common denominator and you can use the output of almost any parser (and you can even reuse the semantic analysis of the closed class words, since these change less than anything from one framework to another). And combining language specific meaning postulates to get from open-class words to domain concepts with domain axioms couched in terms of these concepts makes it possible to divorce the reasoning about the problem from the vagaries of the individual languages. The task is still far from easy, but these steps can make it just a bit less difficult.

## References

- R Crouch, A Frank, and J van Genabith. 2001. Linear logic based transfer and structural misalignment. In H C Bunt, I van der Sluis, and E Thijsse, editors, *4th International Workshop on Computational Semantics*, pages 35–49, University of Tilburg.
- D A Cruse. 1986. *Lexical Semantics*. Cambridge University Press, Cambridge.
- M Dalrymple, J Lamping, F C N Pereira, and V Saraswat. 1996. A deductive account of quantification in LFG. In M Kanazawa, C Piñón, and H de Swart, editors, *Quantifiers, deduction and context*, pages 33–58.
- 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.
- J Hitzeman and M Poesio. 1998. Long distance pronominalisation and global focus. In *COLING/ACL 98*, Montreal.
- M McTear. 1998. Modelling spoken dialogues with state transition diagrams: Experiences with the cslu toolkit. In *Proc 5th International Conference on Spoken Language Processing*, Sydney.
- A M Ramsay and H Seville. 1999. Models and discourse models. In *First workshop on inference in computational semantics*, pages 111–124, Inst. for Language, Logic and Computation, University of Amsterdam.
- H Seville. 1999. Experiments with discourse structure. In H C Bunt and E G C Thijsse, editors, *3rd International Workshop on Computational Semantics*, pages 233–247, University of Tilburg.
- P Tapanianen and T Jarvinen. 1997. A non-projective dependency parser. In *Fifth Conference on Applied Natural Language Processing*.
- M Turunen and J Hakulinen. 2000. Jaspis - a framework for multilingual adaptive speech applications. In *Proceedings of 6th International Conference of Spoken Language Processing (ICSLP 2000)*.