Exploiting the Advantages of Task and Linguistically Oriented Dialogue Management for Robust Interpretation

D. R. Milward

Distribution: PUBLIC

Specification, Interaction and Reconfiguration in Dialogue Understanding Systems
IST-1999-10516

Deliverable D4.4
October, 2002

IST-1999-10516 SIRIDUS

Specification, Interaction and Reconfiguration in Dialogue Understanding Systems

Göteborg University
Department of Linguistics

Linguamatics Ltd

Telefónica Investigación y Desarrollo SA Unipersonal
Speech Technology Division

Universität des Saarlandes
Department of Computational Linguistics

Universidad de Sevilla
Departamento de Lengua Inglesa

For copies of reports, updates on project activities and other SIRIDUS-related information, please look on our website at www.ling.gu.se/projekt/siridus.

For technical matters, please contact the Technical Coordinator, Robin Cooper and for administrative matters, please contact the Administrative Coordinator, Mareike Schmitt.

Prof. Robin Cooper
Department of Linguistics
Gothenburg University
Box 200
SE-405 30 Gothenburg
Sweden

Phone: +46 31 773 2536
Fax: +46 31 773 4853
cooper@ling.gu.se

Mareike Schmitt
European Project Office
Saarland University
c/o EURICE GmbH
Science Park Saar
Stuhlsatzenhausweg 69
D-66123 Saarbrucken
Germany
Phone: +49 (0) 681 959 233 66
Fax: +49 (0) 681 959 233 70
ms@eurice.de

Contents

Chapter 1

Introduction

This deliverable examines different approaches to interpreting spoken utterances, and establishes some of the circumstances which favour the use of grammar based approaches, keyword spotting, or a full semantics. We discuss two novel techniques which have been developed in Siridus. The first is a "semantic chart" which provides a distributed semantic representation. This enables the use of information from keywords, linguistic structure and the information state within a unified framework. The second technique is "semantic-based composition". This uses ontological information to provide connections between concepts, and hence between utterance fragments. The result is a highly reconfigurable and robust interpretation component which allows semantics to propose combinations, not just filter the combinations suggested by syntax.

1.1 Motivation

Most dialogue system designers put a lot of effort into providing prompts which will result in short, simple responses. For these short utterances, the grammar can be compiled into the recogniser with good results. The recogniser will only recognise what is in the grammar, so there is no possibility of it outputting anything other than grammatically well-formed utterances (with respect to the recogniser grammar). Robust interpretation techniques are not needed.

However, system initiated dialogues of this kind can be slow and frustrating for users, inheriting some of the general problems with menu based interaction. Menus are good at guiding the user as to the possible responses required at a particular stage. Where they tend to be poor is when the number of options is large, or the higher level classifications are not

very natural. Consider, for example, deleting a hyperlink in Microsoft Word document. To achieve this users choose the top level menu item, "insert" followed by "hyperlink" followed by "remove hyperlink". Being able to just say "delete the hyperlink" is much more direct and does not require memorising that hyperlink operations are achieved via "insert". A great advantage of language over point and click interfaces is the ability to go straight to what you want rather than traversing a deep menu structure with unclear options at each point. Some advantages of allowing more flexible, mixed and user initiated dialogue are discussed in [5].

Once utterances become longer, and less predictable, it becomes more difficult to encode all possibilities in terms of a fixed grammar. In the ATIS domain for travel booking enquiries, the use of statistical language models based on word n-grams combined with robust interpretation (e.g. [3]) was used very successfully. More recently, in the home control domain, statistical recognition combined with robust interpretation techniques was evaluated against grammar based approaches [4] and gave good results, especially for untrained users. This evaluation did not rely on the statistical recogniser being trained on a huge corpus: instead the system was bootstrapped with an initial 500 line corpus.

1.2 Robust Interpretation

What do we mean by robustness? Menzel [6] suggests that a system acts robustly if:

> a comparatively small deviation from a predefined ideal should lead to no or only minor disturbances of a system's response

This is close to the idea of graceful degradation (see [11] for further discussion). What it means to be a small deviation, or a minor disturbance is somewhat harder to define. In practice, for interpretation, a minor disturbance would tend to be something that would not affect a human's ability to interpret the utterance, or to do something sensible with it.

An aspect of robustness which is not captured by graceful degradation concerns how explicable the behaviour of the system appears to the user. A system which acts in a way which makes sense will appear more robust than one which captures the ideal behaviour more frequently, but also sometimes performs in an apparently random fashion.

Some features which can help to create robust systems include:

1. Avoid bottlenecks e.g. processing passing through a particular stage which cannot be guaranteed. This excludes systems which assume that to get any interpretation

you need a fully connected parse of a sentence, but cannot guarantee such a parse for all hypotheses from the recogniser

2. Distribute processing/representation. The idea here is that if there are independent routes from input to output, failure on one route will not cause all the processing to fail. For example phrase spotting systems which pick out slot value pairings tend to be robust since the each slot value is extracted independently from the input.

3. Use all the information available. If one piece of information is lost e.g. a syntactic connection between two constituents, then the system needs to use other information available e.g. what was likely to have been said in this context. Redundancy means that it is often possible to successfully extract the intended meaning. For example, consider if someone is asked "Where do you want to go" and answers "I want to go to London". Even if "go" or "to" are missed out, "London" is still very likely to be the destination.

1.3 Surface and task based semantics

Exactly what kind of semantics are we aiming to provide from a robust interpretation module? Consider the following possible replies to the question "Where would you like to go?".

1. I would like to go to London
2. I'd like a flight to London
3. I need to go to London
4. I'm going to visit a friend in London

If we try to properly model the semantics of the user utterance we have a relatively hard task. How can we get from "visit a friend in London" to "destination = London"? However, with keyword/phrase spotting we can immediately jump to the correct answer - the question asked about the destination, and part of the user's utterance, London, provides a suitable reply.

Where there is a wide gap between a relatively complicated semantics of the user utterance and what is actually required for the task, keyword/phrase spotting systems may work very well, and have advantages over more linguistically based systems which aim to fully interpret the user input. A traditional linguistically based system needs to provide wide

grammatical coverage to cover what people are likely to say in a domain, and also a substantive inferencing capability to map from the relatively detailed semantics of what users said to something appropriate for the task.

Flight and travel booking are some of the most studied domains in dialogue research, and seem to be good examples where much of user utterances is redundant, and where the user meaning and the task semantics substantially diverge. Phrase spotting thus generally works well, though there are exceptions "Any airport except Stansted" should not be mapped to "Stansted", and "I'd like to go somewhere in Tuscany but not Florence" should not be mapped to "Florence, Tuscany". In other domains there may be much less redundancy, and a closer correspondence between utterance and task semantics. This seems to be the case in the home command and control domain. In an utterance such as "turn on every light in the kitchen", only the words "in" and "the" can be left out without too much effect.

In subsequent chapters we will discuss the use of a distributed semantic representation which enables some convergence between a linguistically based approach and a task based, keyword spotting approach. By using a semantic representation we gain some canonicalisation relative to a surface string of words. Moreover, by using a distributed semantic representation it is also possible to pick out semantic concepts we are interested in just as easily as keyword based approaches pick out words from an input string.

Chapter 2

Dealing with fragmentary and non-fragmentary input

In this section we will try to get a better idea of which current techniques for robust interpretation might succeed or fail on particular inputs, and for particular task semantics requirements. We will look at two different classes of fragmentary input: self contained fragments and dependent fragments.

2.1 Self contained fragments

Consider the following fragmentary utterance:

> I want to go ... let's see ... from London ... to Birmingham

Let us also assume that the required task semantics is in terms of origin and destination e.g.

```
origin_city = London, destination_city = Birmingham.
```

In this case we can imagine mapping independently from each fragment to the corresponding semantics. The result would be a list of slot value pairs. This corresponds to implicitly conjoining the output i.e.

```
origin_city = London & destination_city = Birmingham
```

In this example no special mechanism is required to put together fragmentary utterances. The fragments can be independently mapped to a task semantics, and conjunction used to put together the results. These kinds of example are relatively easy to deal with, either by keyword/phase grammars, or via the collection of information from parsed fragments.

2.2 Dependent fragments

In the previous example, there was no need to put information from fragments back together. However, consider the following example:

> I want to leave Birmingham ... let's see ... at 3 pm

In the travel domain there is the possibility of arrival times and departure times. This means that it is not enough to interpret "at 3pm" as "time = 3pm". Instead, we need e.g. "departure_time = 3pm" or an extra argument which links the time argument to an appropriate event e.g. "time(e1,3pm)". To achieve the correct mapping requires the use of information from two separate fragments. This kind of example is a problem for approaches which use independent phrase grammars (e.g. Nuance SayAnything).

How can we obtain information from separate fragments? Grammar rules only provide a way of grouping material together for well formed utterances. Reconstruction rules provide a way of putting together ungrammatical utterances, provided the errors are predictable e.g. hesitations or self repairs (see the discussion in Deliverable D4.1). What can we do if fragments remain unconnected after syntax and reconstruction?

One possibility is to relax grammar rules, for example, allowing agreement features to mismatch, and to rely on function argument structure and selectional restrictions. This can be combined with "skipping" within the parser e.g. allowing several words to be missed out if they do not contribute to a parse (this approach is described by [10]). This kind of solution will work well if we are very close to a fully connected sentence (e.g. in text based tutoring systems for which this strategy was designed). However, in spoken dialogue, grammaticality can be severely compromised yet what was meant can still be obvious.

Another possible solution is suggested by pattern matching systems. In these systems, separate patterns could be supplied for "at 3pm" in the context of "leave" vs. "arrive". Hand-coded pattern matching rules thus provide a mechanism for glueing together fragments. In later chapters we will look at combining this kind of approach with a more

linguistically motivated approach. The semantic-based composition strategy we describe shows how hand coded rules can be replaced with an approach based on the ontology of the domain.

2.3 Non-fragmentary input

A criticism of keyword/phrase spotting approaches, and, in fact, any system based on shallow linguistic processing, is that the level of analysis is fixed. If it so happens that a full syntactic analysis is achievable, then the information this would provide is not available. Let us consider some examples where this is problematic.

> I want to go from London to Glasgow and from Glasgow to Dundee

In this example, separate treatment of "from London", "to Glasgow", "from Glasgow" and "to Dundee" will leave us with two origin cities, and two destination cities, but no connection between them. In contrast, a full parse and semantics will properly connect "London" with "Glasgow" and "Glasgow" with "Dundee".

A second problematic example concerns bracketing/scope effects. Consider the following example:

> Turn off all the hall lights except the ceiling lights

In this example, we cannot just pick out the key phrases "turn", "off" and "hall lights". We need to distinguish between "the ceiling lights" as the second argument of "except", and "the hall lights" as the first.

Interplay between contextual information, and information within the utterance fragments is neatly exemplified by the following example.

> S: When do you want to leave?
> U1: ... 4pm ...
> U2: Id like to arrive by 4pm

In U1 the recogniser has only picked up the fragment "4pm". In this case it is plausible to assume that "4pm" is the answer to the prior question, i.e. the departure time. However,

if, as in U2, there is evidence of surrounding material talking about an arrival event, this over-rides the context, giving the result "arrival_time = 4pm". The opposite occurs in the next example:

S: Which light do you mean?
U: Is the light in the hall

The extra "is" was inserted by the recogniser since "is the light" has a good trigram score. However, in the context of the question "Which light do you mean", it is more likely that the reply is "the light in the hall". Thus in this case, the context should have preference over material in the utterance. There can therefore be no hard and fast rule to say context should pre-empt utterance content or vice versa. Instead we need to allow for an interplay of preferences, and to use all the information at our disposal, context, syntax and semantics.

Chapter 3

Putting together linguistic and task based interpretation

The robust interpretation module and the dialogue manager together need to provide a suitable action for the system, given an acoustic input and an existing information state. In this chapter we will examine a layered architecture, starting from acoustic input and ending with a system action. We will then look at some of the advantages and disadvantages of the novel approach to providing a robust interpretation component which was described in Deliverable D4.1. This incorporates a task based approach to interpretation within a system which can also use structural linguistic knowledge. A more radical approach is described in the next chapter which allows more interleaving between processing stages.

3.1 From acoustic input to action

In this section we will examine a layered architecture for mapping from acoustic input to a system action. First, let us look at a particularly simple pipelined architecture which uses a speech recogniser to provide a string of words, parses this to create a syntax tree, converts this to a semantic representation, and from this creates a system action:

acoustic input
$\rightarrow_{speech\ recogniser}$
word string
$\rightarrow_{parser}$
syntactic analysis
$\rightarrow_{semantic\ analysis}$

semantic representation
$\rightarrow_{action\ construction}$
system action

We will make four refinements:

1. Remove strict pipelining in the sense of only allowing the input of a process to be the output of the previous process. Instead, we add the output of each stage to the information state. Downstream processes can access earlier output and contextual information via the information state. For example, action construction can access information in the parse tree if it so requires. A similar effect could be achieved with strict pipelining by making each stage add, monotonically, to a single data structure which is then passed to the next stage.

2. Provide multiple analyses at each stage. This avoids early commitment by processes which only have access to a limited amount of information. For example, the output of the recogniser is an n-best list, or a word lattice that represents compatible lexical hypotheses in an efficient packed format. This allows for later processing to choose an initially dispreferred hypothesis if there is strong justification from domain knowledge, contextual or syntactic information.

3. Split the semantic analysis stage into two, first the determination of what the user meant. Secondly, the determination of what this means in the context of the task. For example, in mapping from an acoustic input corresponding to "turn on the bedroom light", to a message sent across a home network consisting of the string "house control on device_1234" the system might use a task semantics such as "change-state(device_1234,on)".

4. Allow for partial analyses. For example, rather than a set of fully formed semantic representations we will provide a set of semantic edges (a semantic chart) which encodes partial semantic analyses, and full analyses if these are available.

The stages are now as follows:

acoustic input
$\rightarrow_{speech\ recogniser}$
word lattice
$\rightarrow_{parser}$
constituent edges
$\rightarrow_{utterance\ semantic\ analysis}$

semantics edges for user utterance
$\rightarrow_{task\ semantic\ analysis}$
task based semantics
$\rightarrow_{action\ construction}$
new context & system action

In this model, a later stage of processing does not need to use the output of a previous layer at all. For example, task semantic analysis might want to bypass utterance semantics and the parser output if there is adequate evidence from the context and particular words expressed, but the actual user utterance meaning is obscure.

It is worth noting that we cannot assume that there will always be enough information to provide a fully specified task semantics. Further clarification may be required. For example, if there are two bedroom lights, the system might want to first ask for clarification i.e. "which bedroom light do you mean, the standard lamp or the bedside lamp". Thus the system action may be an enquiry to the user, rather than a command or query of a device or a database.

3.2 Bridging the gap between task based and linguistically based systems

In the architecture described above, there are two alternative places where you might expect a robust interpretation module. Firstly, in providing a task based semantics directly from a word lattice (similar to keyword spotting). Secondly, in providing a connected (or more connected) user semantics.

In this section we will discuss a third alternative. Since semantics provides a more canonical representation than a word string, rules mapping from user semantics to task semantics should be preferable to rules working directly on words or phrases. To achieve this we need a new approach to semantic representation. Traditional connected semantic representations (e.g. logical forms) are not ideal for "pattern matching" approaches. It is much easier to pick out individual words e.g. "arrive" and "at 3pm" from a word string, than to pick out the equivalent semantic items from a logical representation. However, it is possible to convert from a traditional semantic representation into an indexed representation, where items are connected indirectly via index equality. This more distributed representation is much more convenient for rules which pick up individual pieces of semantic representation.

The work described in detail in [7] and [11] uses a semantic chart. This is an indexed

representation which allows packing similar to a syntactic chart. The chart is constructed by adding indexed concepts, and relationships between indices, on top of the word lattice. Mapping rules (from semantics to slot-fillers) apply directly to the semantic chart. In being able to use structural and contextual information where it is available and relevant to the task, the approach was able to improve on keyword or phrase spotting approaches.

3.3 Successes and limitations

Certain aspects of the work described in [11] seemed very successful:

1. The use of a distributed semantic representation allowed access to pieces of semantic representation even if they were already connected with a larger structure. Thus information within one fragment of an utterance could influence the interpretation of another fragment.
2. The use of a semantic chart provided a uniform method for packing analyses, from the lattice right upto the semantics. Again this seemed successful.
3. The use of different sources of information, keyword, context and syntax were shown to improve performance.

The mapping rules had some advantages over simple pattern matching:

1. There is a single strategy of "most specific rule wins". This avoids the need for explicit rule ordering.
2. Linguistic structure is available to allow more fine grained rules

However there are also some disadvantages when compared to a traditional full semantics approach:

1. the rules are hand coded
2. the rules are not compositional

The semantic-based composition strategy described in the next chapter is aimed at addressing these defects to provide a more easily reconfigurable system.

Chapter 4

Semantic-Based Composition

To use semantics for composition we need firstly domain knowledge, and secondly a strategy for combination. The first section discusses an example ontology for the domain of controlling and querying networked home appliances. The next section considers a simple combination strategy, and how this might be used to reconstruct a fragmented semantic representation. A more radical approach of "semantic-based composition" is described in the final section, which allows for semantics to propose combinations in parallel, or even in competition with syntax.

4.1 Incorporating domain knowledge

We need to provide domain knowledge which will allow semantics to propose combinations. An ontological description provides a good starting point, providing knowledge about a domain and some limited, but tractable inferencing capability (see e.g. [8] for comparison between ontologies expressed in description logic vs. FOL). As an example, consider a domain ontology for the home network application. The following "is-a" hierarchy provides relationships between concepts down to individual devices and rooms. An "in" hierarchy states which devices are in particular rooms, and which rooms are in the house. The "is-a" and "in" relations are both transitive (a bedroom is-a location, l1 is in house1). Other non-transitive relationships may be required e.g. "b1" has position "back".

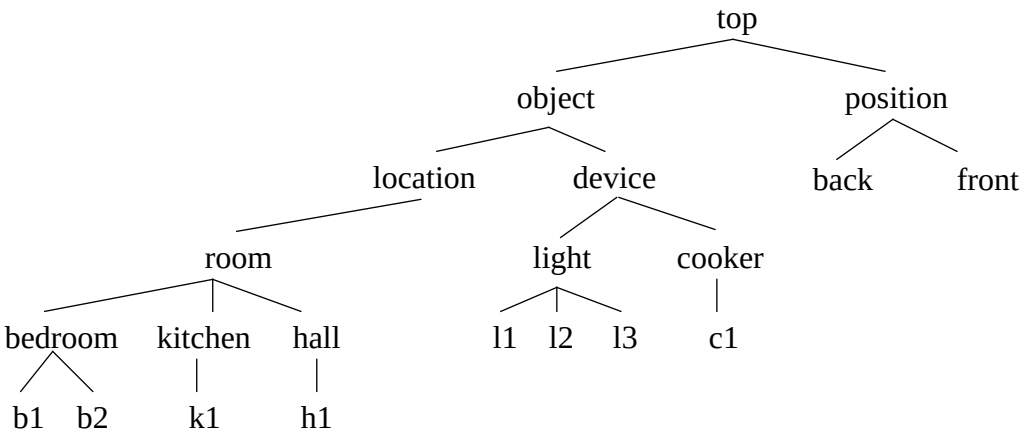

Figure 4.1: Part of the is-a hierarchy for the home domain

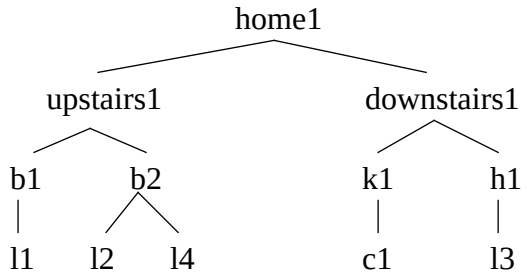

Figure 4.2: Part of the in hierarchy for the home domain

4.2 Using semantics for fragment combination

In this section we explore a reconstruction approach based on ontological knowledge. Consider the following example:

Turn off the light ... bedroom

To combine "bedroom" with "light" we can use general information that a bedroom is a location, and a light is a device, and that devices have a location, or we can use specific extensional information that in this particular house there is a light in a bedroom. However, performing the combination itself is non-trivial. "Turn off the light" is a well formed syntactic/semantic fragment. How can we now add in "bedroom"? A distributed representation makes this easier, since at least we can isolate and access the index for "light" which we can then associate with "bedroom". For example, it is possible to imagine a generic procedure which replaces the index for light with the index for "light in bedroom".

turn_off(i1), i1:the(i2), i2:light, i3:bedroom
→
turn_off(i1), i1:the(i4), i4:in(i2,i3), i3:bedroom

This analysis may seem superficially plausible, but "i3" is not actually of the right type to be an argument of the "in" relation. The "in" relationship is between instances of devices and instances of rooms, not between instances of devices and a kind of room. To correct this requires the addition of a quantifier to associate with "bedroom". Possible choices would correspond to "in a bedroom", "in the bedroom", "in any bedroom", or "in every bedroom". It is not clear what the correct choice would be, and it may also be irrelevant in a particular scenarios (e.g. if there is a light in only one of the bedrooms, or if there is only one bedroom).

A similar example concerns cases where there may be more than one semantic relationship between two fragments which are unconnected in syntax. Smoke alarms provide an interesting case. A smoke alarm for a kitchen may be located outside the kitchen. Thus we need to provide a "for" relationship as well as an "in" relationship. Now consider:

turn off the smoke alarm ... bedroom

Let us assume that the bedroom smoke alarm is both in the bedroom, and for the bedroom. Do we now have to choose one relationship or the other for the reconstruction?

Although it is possible to use underspecification techniques to avoid making explicit choices within a reconstructed semantic representation, these examples suggest that it might be cleaner to use semantic knowledge more directly, and to avoid performing explicit reconstruction. To do this is the task of the next section, and requires a rather different way of thinking about syntactic and semantic structure.

4.3 Semantic-Based Composition

In this section we will describe an approach to combining items together using domain knowledge. There are no hand coded rules. Instead, ontological information is used to provide connections between concepts, and hence between utterance fragments. The semantic-based composition can be used to connect fragments left over after syntactic combination, or, perhaps more interestingly, in parallel or even in competition with syntax.

In the last chapter we proposed a monotonic accumulation of information from word hypotheses upto semantics. Here we adopt essentially the same structures using distributed representations for syntax and semantics, but take a different perspective. In the previous sections, semantics was always driven by syntax, with mapping rules providing some limited semantic composition. Under the new perspective we allow semantics to do far more work. Lexical items are associated with concepts, and relationships are established between the concepts. We end up with a web of inter-connected concepts, with concepts raised in "salience" according to the amount of inter-connectivity. The feel of the system is thus close to that of cognitively inspired approaches [2], where lexical and encyclopaedic knowledge is contained within the same network. However, the approach is still designed around conventional syntax and semantics. The indices provided by the distributed representation provide the nodes in the network.

In the implementation we will describe next, we actually use no syntax at all. The input string is effectively treated as an unordered bag of words, except for multi-words such as "hi fi". This gives a good idea of how successfully semantics can be used for composition. The resulting system has advantages over keyword based approaches in not requiring hand coded rules, and in allowing recursive combinations. Note that the lack of syntax and word order is certainly not intended as a permanent feature. Syntax is a very good clue, often the only clue for establishing proper connections. The intention is that the final system will use all the possible sources of information including syntactic constraints, and contextual constraints.

Consider the following example:

```
back ... bedroom ... light
```

The initial layer consists of the words themselves.

```
w1: back   w2: bedroom   w3: light
```

These are linked to the appropriate concepts:

```
c1: back   c2: bedroom   c3: light
```

For each concept the system forms links to the instances e.g.

```
c1: back   c2: bedroom   c3: light
           b1 b2         l1 l2 l3 l4
```

For each instance, the relationships between that instance and other instances or concepts are considered. For example, there is an "in" relationship between "l1" and "b1" and between "l2" and "b2".

In the first pass, b1 is the most salient instance, since it is related both to "back" and to "l1". In the next pass, salience is percolated through, so "l1" gains salience from "b1". The result is that the two most salient entities are the bedroom (which is in the back and contains a light) and the light (which is in a bedroom at the back). Thus we obtain the back bedroom light as required.

If syntax had been available e.g. from a well formed "back bedroom light" the correct analysis would be reinforced. The syntax tells us that there is some relationship between "bedroom" and "light", and some relationship between "back" and either "bedroom" or "light". The "back bedroom" reading would be given preference due to the ontological relationship between "back" and "bedroom", but not between "back" and "light".

4.4 Evaluation

The evaluation domain chosen was command and control of networked devices e.g. "turn off all the lights in the kitchen" or "have I left anything on?". 50 utterances were selected at random from the GeneralX corpus described in [9]. We then examined the performance of the semantic-based composition system by hand. The results were as follows:

- incorrect: 1

- partial: 4
- correct, but: 20
- correct: 25

The evaluation classification was based on the following criteria:

1. incorrect: response is incorrect or useless
2. partial: something is correct, but not a complete response
3. correct, but: strictly speaking correct, but could be more helpful
4. correct: correct and a reasonable response

The number of incorrect responses was surprisingly low, given that no syntax or word order is used. The one incorrect response was from:

> put the stove and switch off the kitchen light

The system response was to switch off the stove. Problems with conjunctions were to be expected since word order is needed to associate the correct words with each conjunct. The partial results were also due to conjunctions, with the system only querying e.g. one device when more than one was specified.

The "correct, but" cases are somewhat mixed. 9 of these were where the system said it did not understand descriptions of devices which did not exist in the current house. This is sensible behaviour, though it might be more informative to say "I didn't recognise the device you wanted to turn on". In 4 cases the system said it did not understand, but it should have done: these were due to inadequate coverage. 5 were cases where the system was overinformative e.g. "What is in the kitchen" got the reply "the kitchen light is on, the cooker is on ...".

This was a relatively small evaluation, but does suggest that semantic-based composition provides a useful strategy. In most cases semantics has enough information to put words together to create correct commands and enquiries. However, as expected, semantics is not enough for sentences involving conjunctions, which bring in a need to recognise bracketing and scope (for 10

4.5 Comparison

In semantic-based composition, ontological information is used to elaborate the original input material. User utterances can be properly generative: there is no fixed set of slots to fill. It thus differs from "script" based approaches where input values fill a pre-defined structure.

A somewhat similar approach is adopted by Abella and Gorin [1] who use an inheritance hierarchy of call types to combine the results of a task based language understanding module. In this system the results are combined not just by conjunction, but also by disjunction, according to the whether they make sense as separate requests or as mutually exclusive alternatives. The use of the hierarchy to guide composition is limited to combining 'propositional' fragments via boolean operators.

Sadek et al.[12] use a semantic network, and do allow combinations of concepts according to arbitrary relationships within the network. They assume "semantic connectiveness" i.e. a user never mentions two concepts without (at least implicitly) linking them by some relation or other concept. The aim of "semantic completion" is to find the best path through the concepts within a semantic network.

Both these approaches use the semantic information at a second stage of processing after syntax. For example, Sadek et al. perform an initial island parse to create the list of concepts to send to semantic completion. There is no possibility of syntactic relationships reinforcing or competing with semantics: once a list of concepts is produced from the parser, syntax has no further role. This means that both approaches suffer from the criticism we made earlier of shallow approaches: even if there is a completely connected syntactic representation this information will not be used.

Finally, we should compare with the rather different approach of Rose [10]. Here there is no separate ontology, but some semantic information is encoded within selectional restrictions. For example, the word "back" might require an argument of type "room", allowing "back" and "bedroom" to combine. However, function argument structure only seems to provide part of the information needed for semantic combination. "kitchen" and "light" are not related via one being a function, and the other its argument. Instead, it is the implicit relation "in" which has both as its arguments.

Chapter 5

Conclusions

The work in Siridus on robust interpretation aimed to provide a bridge between task oriented and linguistically oriented approaches. This has definitely been achieved. The work in D4.1 provided a system which enabled linguistic information to be combined with task-based keyword spotting. What was missing was the compositional nature of a true linguistically oriented approach, and the advantages this has for reconfigurability. This is what we hope to have achieved with the semantic-based composition strategy described here. Composition can occur, even if syntactic clues are unavailable, with no hard coding of composition rules. As a side effect we have provided another bridge, from linguistically oriented approaches to approaches from the tradition of cognitive linguistics.

Bibliography

[1] Alicia Abella and Allen Gorin. Generating semantically consistent inputs to a dialog manager. In *Proceedings of Eurospeech*, 1997.

[2] Richard Hudson. *English Word Grammar*. Oxofrd: Blackwell, 1990.

[3] S. Isaar and W. Ward. CMU's robust spoken language understanding system. In *Eurospeech*, pages 2147–2150, 1993.

[4] Sylvia Knight, Genevieve Gorrell, Manny Rayner, David Milward, Rob Koeling, and Ian Lewin. Comparing grammar-based and robust approaches to speech understanding: a case study. In *Proceedings of Eurospeech*, 2001.

[5] Staffan Larsson, Gabriel Amores, Elena Karagjosova, David Milward, and Dimitra Tsovalzi. Flexible dialogue. Technical report, Siridus Deliverable D1.4, 2002.

[6] Wolfgang Menzel. Robust processing of natural language. In *Proceedings of the 19th German Conference on Artificial Intelligence*, 1995.

[7] D. Milward. Distributing representation for robust interpretation of dialogue utterances. In *Proceedings of the 38th ACL*, pages 133–141, Hong Kong, 2000.

[8] D. Nardi and R. J. Brachman. An introduction to description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logic Handbook*, pages 5–44. Cambridge University Press, 2002.

[9] J.F. Quesada, J.G. Amores, P. Manchon, G. Perez, S. Knight, D. Milward, and J. Thomas. Possibilities for enhancing speech recognition by consulting information states. Technical report, Siridus Deliverable D2.3, 2002.

[10] Carolyn Rose. A framework for robust semantic interpretation. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics, NAACL*, 2000.

[11] C.J. Rupp and David Milward. A robust linguistic processing architecture. Technical report, Siridus Deliverable D4.1, 2000.

[12] M.D. Sadek, A. Ferrieux, A. Cozannet, P. Bretier, F. Panaget, and J. Simonin. Effective human-computer cooperative spoken dialogue: The ags demonstrator. In *Proceedings of the Fourth International Conference on Spoken Language Processing (ICSLP)*, 1996.