

Semantic representations and robust interpretation in the HIGGINS spoken dialogue system

Gabriel Skantze¹

`gabriel@speech.kth.se`

This paper discusses the different requirements that should be put on the interpretation component in a dialogue system capable of understanding spoken language. It is argued that such a component should be able to, among other things, find combination of phrases and sub phrases, accept insertions in the input, accept and analyse non-congruence and adequately interpret self-corrections. The paper describes the techniques utilized the HIGGINS spoken dialogue system project in order to meet these requirements.

1 Introduction

An important bottleneck for many dialogue systems built today is the speech recogniser, which tends to introduce miscommunication in the human-computer dialogue. These kinds of error should be prevented, detected and handled by the dialogue system in a way that maximizes the user's satisfaction of using the system. The first step in the error handling process is to make robust and adequate interpretations of the speech recognition result. This paper will describe how such robust interpretations are achieved in the HIGGINS spoken dialogue system.

HIGGINS is a newly started project at KTH about error handling in spoken dialogue systems. To be able to implement and test different techniques for error handling and robustness, we will develop a spoken dialogue system. The primary domain for the system is navigation for pedestrians. Later on, a secondary domain for HIGGINS will be simple guidance (e.g. "where is the closest restaurant") and information about objects in the environment. Thus, we have to take the representation of such utterances into consideration at an early stage. The results from the project should be applicable to similar domains, such as car navigation and museum guides. The domain is chosen because it will probably give rise to different kinds of interesting error types and error handling strategies, but it also requires advanced semantic representations which may be challenging when it comes to robustness. As a first step, a modified version of the Wizard of Oz experiment was conducted, where an ASR was used to introduce errors that are typical for dialogue systems (Skantze, 2003). The collected dialogues and error situations that have been identified has been a starting point for the development of the dialogue system.

This paper will first discuss the problem of interpreting spoken language and the requirements for the interpreting components in a spoken dialogue system that should allow robust interpretation. The domain for the HIGGINS spoken dialogue system and the planned architecture will then be discussed in more detail, along with the requirements that the HIGGINS domain puts on the dialogue system. This will be followed by deeper discussions of the components that are relevant for robust interpretation and knowledge representation.

¹ This report is based on the work by Gabriel Skantze, Jens Edlund and Rolf Carlson in the HIGGINS project.

We will start by describing how the domain model is represented in the database and how this model can be queried. We will then show how underspecified world descriptions can be enriched so that they can represent the semantics of utterances. Finally, a parser that can generate these semantic representations from spoken natural language utterances will be described.

2 Interpreting spoken language

2.1 Uncertainty in spoken language

One of the major problems with the semantics of natural languages is that of uncertainty and ambiguity. The issue of semantic representations is a general problem in the field of natural language processing, which has to be handled in most NLP applications. However, spoken dialogues seem to have some properties that make traditional interpretation techniques (which have been developed for written language) problematic to apply.

According to Heisterkamp et al. (1992) dialogue semantics has to deal with three sources of uncertainty. The first is the general ambiguity of language. Each word or sentence can often have several different meanings. The interpretation of an utterance also relies on the knowledge of the agent. A second problem inherent in dialogue is that the interpretation of an utterance often is heavily dependent on context. Anaphora, ellipses and deictic expressions are common in dialogue, as the speakers are trying to reduce the effort in forming their utterances and instead rely on the common ground shared with the other speaker. The interpretation of such expressions is dependent on the textual or non-textual context. A third source of uncertainty is specific for spoken dialogue systems. The coded utterance, represented as a textual string, that is to be interpreted is the result of an error-prone speech recognition process. This means that the resulting representation may contain errors that are independent of the parsing algorithms. The problem here is not only how to make the correct interpretation, but also how to represent the uncertainty that comes from the noisy input. The dialogue manager must deal with hypothetical interpretations that may be proven wrong.

2.2 Fragmentary input

In spoken dialogue, utterances do not necessarily take the form of complete sentences. Instead, utterances often consist of fragments, filled pauses (like “ehm”), pauses, etc, which all may have meaning in the context. This somewhat unpredictable behaviour makes it hard to apply grammatical frameworks, developed for idealized written language, to spoken language. Poesio (1996) calls these units micro conversational events. In order to make adequate interpretations of utterances, such events should be handled as well, not just be seen as noise in the input. Often, such utterances have interactive functions.

Utterances can also be fragmentary due to disfluences. For example, one could make a self-correction in the middle of an utterance, such as “I want an apartment on Vasaväg... Valhallvägen”. Topicalizations are also common, such as “The green apartment ... what does it cost?” (Bell et al., 2001). Silent pauses in the middle of an utterance pose a problem. Is the utterance complete, or should the system wait for more input? This means that we should consider some sort of incremental parsing, i.e. interpret the utterance during the production. One could also consider letting the system take actions during the production, such as eliciting feedback or making interruptions. The problem of fragmentary input could be seen as just a parsing problem; however it is also important to be able to represent fragmentary input, if the system is supposed to take actions in the middle of an utterance. Bell et al. (2001) argues that utterances should be incrementally parsed and represented as closing or non-closing. If an utterance is determined as non-closing, the system should wait for more input, but it should also signal continued attention to the user by for example using facial gestures such as lowering the forehead, looking at the user. Since the utterances are incrementally

parsed, there is also need for representation of fragments, which can be unified with the representation of later utterances.

2.3 Requirements for interpreting spoken language

The previous discussion shows that a robust interpretation component in a dialogue system for spoken language has some special requirements that have been considered. Such a component should support, among other things:

- Interpretation of results from a speech recogniser with stochastic grammars. Such grammars may give “non-grammatical” results (compared to context-free grammars), but they may also better cover what people actually say (since people normally do not speak “grammatically correct”). When errors occur in such recognitions, the string that the parser gets can be very noisy.
- Depending on the domain, the parser should use morphological information to distinguish for example nominal phrases with different number and definiteness. This information may be distorted by the speech recogniser, due to acoustic similarity, and may result in non-congruent phrases, which should still be interpreted. For simpler domains, such as travel information, morphological features may not be important, but for a domain such as HIGGINS they may, as will be shown later, be very important.
- The parsing of sentences, phrases and keywords, depending on how well the grammar can cover the noisy input. The parser should deliver the solutions that cover the most of the input string. If it fails to find full phrases, it should fall back on sub-phrases and then on keywords in order to come up with a guess as good as possible. Simple keyword spotting is not enough for a domain that is as complex as HIGGINS.
- Robust and general parsing rules. This means that each rule should not be manually built for all sorts of deviations that may occur in the noisy input. The parser should instead automatically handle both unexpected insertions in the input (which may be caused by filled pauses and disfluences), as well as non-congruent phrases, without having to explicitly write this into the rules. The parser should also utilize knowledge of how the domain is structured when unifying the semantics of a phrase, in order to make the rules more robust and general.
- Incremental parsing. The parser should incrementally deliver results in real time as the user is speaking. In this way, other components than the speech recogniser can determine, for example, when the system should start to speak. The system could also elicit feedback while the user is speaking, without using the dialogue manager. The user must in that case be able to make self corrections.
- Parsing confidence scores. The parser should use the word confidence scores that the speech recogniser delivers in order to give the hypothesis confidence scores. The computation of such a confidence score should also depend on insertions and non-congruence in the input.

This list is by no means exhaustive, but it will be the basis for the following discussion.

² I.e. not in line with the normative grammatical rules developed for written language.

3 The HIGGINS spoken dialogue system

3.1 The HIGGINS domain

The primary domain for HIGGINS is navigation for pedestrians. The user is standing somewhere in a city and wants to go somewhere. The dialogue system helps her in solving the task. Since the system does not know where the user is, it has to rely on the user's descriptions of the environment.

One of the reasons for choosing the domain is that it needs a fairly advanced semantics in order to represent the user's utterances. The general approach to robustness in interpretation of speech recognition results has been to use keyword spotting. This works fairly well for more simple domains, but will not work for the HIGGINS domain, since the user's descriptions of the environment may contain complex relations. Morphological information must also be used in order to make distinctions between for example singular-plural and definite-indefinite. The utterance "I am standing at the building" has for example a very different meaning than "I am standing at a building", since it probably refer to a different building.

In order to test the dialogue system, the user must be able to move in a three dimensional world. Real world user testing would of course be the ultimate test, but would be too costly during development, and we would lack control over the modelled environment. Instead, we will use VRML – a standard markup language for 3D modelling – for which there are a lot of available browsers and tools. The VRML code can be automatically generated from the dialogue system domain model, which has the advantage that we do not have to build the domain model based on reality, but instead vice versa. This ensures that the reality and the model of it agree. An example of a VRML scene rendered in a VRML browser is shown in Figure 1.



Figure 1: An example VRML scene built for HIGGINS user testing.

An example of a possible dialogue with HIGGINS is shown below:

User: I want to go to the closest subway station.

System: Ok, to the closest subway station. Can you describe where you are now?

User: I have an ATM to my left and a pedestrian crossing in front of me.

System: Can you see some trees to your right and a white building in front of you?

User: Yes

System: Ok, take left after the large building which you have on your left and follow the street until you reach a crossing.

User: Ok, there is a bus station here.

System: That's right. Take left again after the bus station.

In order to make this kind of dialogue possible, the dialogue system must be capable of (among other things):

- Representing complex properties of objects as well as (especially spatial) relations between objects. Nested relations should also be possible to represent (such as “I am standing next to a tree which is standing on a lawn”).
- Morphological distinctions, especially between singular/plural, definite/indefinite.
- Engage in a dialogue to find out where the user wants to go and then iteratively updating a hypothesis of the user's position based on the user's descriptions of the surroundings.
- Compute the user's position using spatial and temporal reasoning.
- In case the user's descriptions of the environment are not sufficient for determining the position, the system should engage in a dialogue about this.
- Generate route directions in appropriately sized chunks. This also includes finding the optimal path and to generate descriptions using grounded concepts.

3.2 Dialogue system components

The aim of the HIGGINS project is not to build a general framework for developing dialogue systems such as TrindiKit (Larsson, 2002), LinLin (Jönsson, 1997) or Atlas (Melin, 2001). Instead, a set of (possibly) reusable components with clearly specified input and output will be implemented. One research question is how each of these components should handle errors and make the dialogue system more robust. The components will communicate over a broker³ using XML, which means that each component can be implemented in any language and can be included in a distributed architecture. The components do not need to know in which context they are used. This kind of architecture has previously been used at KTH in for example the AdApt project (Gustafson et al., 2000).

In order to understand where the components described in this paper fit into the dialogue system, a rough sketch of the planned dialogue system architecture is shown in Figure 2.

³ A server which forwards function calls, results and error codes between program modules over the Internet.

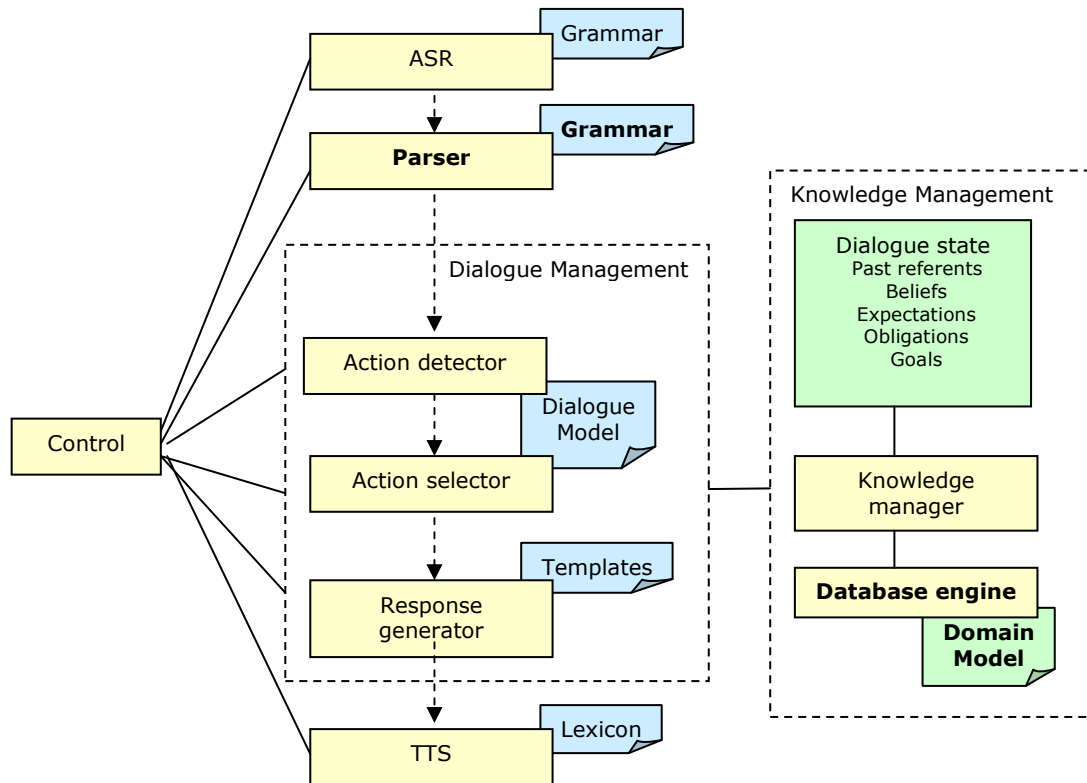


Figure 2: The planned architecture of the HIGGINS dialogue system. The components that are discussed in this paper are written in bold.

- The **control** component is just concerned with communication between the components that constitutes the dialogue system. The control module knows which components are plugged into the system and mediates most communication between the other modules. This makes it possible for the other components to work in an asynchronous fashion.
- The **speech recogniser** must be able to deliver word confidence scores. We will use TMH's Daytona speech recogniser with stochastic n-gram grammars.
- The **parser** takes the output from the ASR and returns a set of (possibly underspecified) semantic representations of the utterance without taking the context into consideration. It is described in section 7.
- The **dialogue model** includes rules for the system's goals, expectations and obligations. It will be used for guiding the interpretation of the user's utterance and selecting the next system action.
- The **action detector** takes the output from the parser and situates it in the context (using the dialogue state and dialogue model) and returns the actions that the user really is trying to perform with the utterance. It should also instantiate anaphoric references using the dialogue state.
- The **action selector** takes the user's action and decides the next system action using the dialogue model and its current beliefs, expectations, obligations and goals. The output will be a semantic representation of the system's action.
- The **response generator** will take the semantic representation of the system's action and generate a text using a set of templates. It is also possible to run the parser "backwards" in order to generate strings from semantic representations. This could

be used to “paraphrase” what the user has said (see section 8.5 for an application of this).

- The TMH speech synthesis will be used for **text-to-speech**.
- The **knowledge manager** handles complex (possibly ambiguous) requests for information about the world and the ongoing interaction. This includes reasoning about the domain. The knowledge manager performs the necessary transformations on the semantics in order to access the database engine(s). It also decides which database to access. The knowledge manager is highly domain dependent and will probably have to be programmed specifically for the domain. The use of a knowledge manager, which the dialogue manager interacts with, is inspired by Flycht-Eriksson (2001).
- The **database engine** handles simple (disambiguated) requests to the domain model. It is described in section 5.
- The **domain model** is a large tree structure represented in XML. This is described in section 4.
- The **dialogue state** includes past discourse (for reference resolution) as well as the system’s current beliefs, expectations, obligations and goals.

3.3 Using XML

We have chosen to use XML (XML Core Working Group, 2000) for all descriptions in the dialogue system: resources, configuration and communication between components. XML is simply a standard for describing tree structures that has some notation for making references between nodes so that cyclic graphs can be represented. Tree structures are widely used for computational representations. Given that a tree structure is suitable for the representation, XML has the following advantages:

- XML is a standard and a de-facto standard.
 - XML Schema is standard that makes it possible to type XML, give it default content and validate it (XML Schema Working Group, 2001).
 - XSLT is a standard for making transformations on XML structures and convert them to other structures for different purposes (XSL Working Group, 1999b).
 - XPath is a standard for extracting content in an XML structure (XSL Working Group, 1999a).
- There are a lot of tools and API:s for handling XML for various platforms and programming languages.
- For simple structures, plain XML is easily readable. This advantage decreases when the structures grow more complex, but by using style sheets, even complex XML structures can be presented in various ways depending on what one wants to study.

The disadvantage with XML is that it can be “wordy” which could possibly make it slow to process and sometimes awkward to write by hand.

XML will be used for all kinds of representation in HIGGINS. To be able to mix XML entities that are used for conceptually different things in the dialogue system and that are specified using different schemas, we use XML namespaces everywhere. The namespaces that are used in the examples in this paper are shown below:

xs xml scheme Entities related to the XML Schema specification.

w	world	Description of the world. Domain-related propositional semantics.
de	database engine	Database engine communication.
s	semantics	Utterance semantics.
g	grammar	Grammar markup.

4 Domain model

To be able to talk about the world, the system must have a model (representation) of it. The basic entities in the HIGGINS domain model are objects, represented as trees, which have subordinated properties nodes. Relations between objects are treated as properties of the objects and are thus not top nodes. This has the advantage that it is fairly straightforward to express complex chained relations (such as “I am standing next to a tree which is standing on a lawn”), which can be embedded in a single object representation. This way of representing relations also lies closer to the structure of natural language, which means that they can more easily be built in the parsing process. The drawback is of course that there are redundancies – the relation is represented in each object that is part of the relation. However, the actual representation in the database can be more efficiently implemented; this representation of relations is mainly chosen to make queries to the database and representations of utterances more convenient.

An object has the following specification:

id	Unique identification of the object
type	An object class, which both constrains and gives default values for the content of the object.
shape	A description of the geometrical shape of the object (seen from the above). It can be a point, a rectangle, a line or a polygon. This description also include the shape’s coordinates and height without reference to its placement in the world
properties	Inherent properties of the object, such as material, size and colour, which can be talked about.
position	Position of the object in the world. This includes both the object’s coordinates in the actual world and it’s rotation. The position description also includes the object’s spatial relations to other objects (direction and distance).
subobjects	Other objects that are parts of the object. For example, a house may have walls and a street may have street segments as sub objects.
references	References to other objects that have a reference to this object. For example, a crossing may have references to the intersecting streets and street segments may have references to the streets that they are part of.

A typical description of a concrete building could look like this:

```
<w:object xs:type="building" id="2">
  <w:shape xs:type="rect">
    <w:width>560</w:width>
    <w:length>105</w:length>
    <w:height>50</w:height>
  </w:shape>
  <w:properties>
    <w:material>concrete</w:material>
  </w:properties>
  <w:position>
    <w:rotation>75</w:rotation>
    <w:coords>1552 2060</w:coords>
  </w:position>
</w:object>
```

All measures are given in decimeters.

A user is also modelled as an object. However, the system will probably never have a model of the user's absolute position, but only her spatial relation to other objects, based on her verbal description of the environment:

```
<w:object xs:type="user">
  <w:shape xs:type="point"/>
  <w:position>
    <w:rotation>45</w:rotation>
    <w:relation>
      <w:object xs:type="ref" id="2"/>
      <w:direction>80 100</w:direction>
      <w:distance>40 80</w:distance>
      <w:niveau>0 0</w:niveau>
    </w:relation>
    <w:relation>
      <w:object xs:type="ref" id="29"/>
      <w:direction>260 350</w:direction>
      <w:distance>30 50</w:distance>
      <w:niveau>0 0</w:niveau>
    </w:relation>
  </w:position>
</w:object>
```

Directions are given in degrees (starting from north, counting clockwise). Notice the under-specified distance and direction to the objects, represented as spans.

In the XML description, the related objects are only referred to using the object id. In the internal representation of the tree, this reference is replaced by an actual pointer to the object, which forms a cyclic graph, so that the descriptions of the related object can have an infinite depth. This makes it possible to represent complex facts. For example, consider the representation of the fact the user has a large tree to the left, which is standing on a lawn:

```
<w:object xs:type="user">
  <w:shape xs:type="point"/>
  <w:position>
    <w:relation>
      <w:object xs:type="tree">
        <w:properties>
          <w:size>large</w:size>
        </w:properties>
        <w:relation>
          <w:object xs:type="lawn"/>
          <w:niveau>-1</w:niveau>
        </w:relation>
      </w:object>
      <w:direction>250 290</w:direction>
    </w:relation>
  </w:position>
</w:object>
```

Niveau denotes the spatial relation in the vertical plane. -1 means below, 0 the same level, and 1 above.

Except for being used as a database, the domain model can easily be transformed using XSLT to Scalable Vector Graphics (SVG) to render 2D-maps of the world, as well as VRML (as mentioned before) to render a 3D-model of the world that can be used for user testing.

5 Database queries

The domain model is used by the database engine to perform queries about objects in the world and their relations. The basic database query is a search for objects in the world. If some specific property or relation of this object is requested, this can easily be derived from the response object using XPath. The search query consists of an underspecified object, which the database engine tries to unify with all objects in the database. The objects that unify successfully are returned fully specified. The unification is really a unification of tree structures with some support for logical operands such as “and”, “or”, “xor”, “less than”, “more than”, regular expressions and intervals. The use of underspecified objects for queries makes the step from database queries to the representation of the propositional content of utterances fairly straightforward, which is shown in section 6. Some examples of database queries are shown below:

<p>Give me all large trees that are close to a concrete building.</p>	<pre> <de:query> <w:object xs:type="tree"> <w:properties> <w:size>large</w:size> </w:properties> <w:relation> <w:object xs:type="building"> <w:properties> <w:size>concrete</w:size> </w:properties> </w:object> <w:distance>0 50</w:distance> </w:relation> </w:object> </de:query> </pre>
<p>What can the user see at position 1975, 827, facing east?</p>	<pre> <de:query> <w:object xs:type="user"> <w:position> <w:rotation>90</w:rotation> <w:coords>1975 827</w:coords> </w:position> </w:object> </de:query> </pre>
<p>Give me a user position where there is a wooden building to the left and a tree to the right.</p>	<pre> <de:query> <w:object xs:type="user"> <w:position> <w:relation> <w:object xs:type="building"> <w:properties> <w:material>wood</w:material> </w:properties> </w:object> <w:direction>250 290</w:direction> </w:relation> <w:relation> <w:object xs:type="tree"/> <w:direction>70 110</w:direction> </w:relation> </w:position> </w:object> </pre>

</de:query>

The search for possible user objects (as in the last example above) must be treated in a special way, since all possible user positions cannot be enumerated and stored in the database. Thus, the unification of tree structures has been enhanced with some reasoning about possible user position and how these relate to objects in the world, depending on the user rotation. This, however, is transparent to the client using the database engine (which will be the knowledge manager). The result of such a search is a position on a node in the street network or a street segment between two nodes in the network. When the system can determine on which node or between which nodes the user is (and possibly in which direction the user is standing), it can also give route directions towards the goal. The result from the database engine will also contain everything that can be seen from every possible user position. This can be used by the system to pose questions to the user in order to determine the user's position if there are several possible matches.

6 Utterance semantics

The representation of objects and queries described so far covers the propositional content of utterances, but they are not rich enough to represent the complete semantics of utterances.

6.1 Given/New

Unlike many other simpler domains, a dialogue system about objects and their relations in the user's surrounding needs to capture the distinction between given and new, i.e. what has been talked about before and what is introduced in the utterance (cf. Brown & Yule, 1983). For example, the utterances "the large building is to my left" and "a large building is to my left" have quite different meanings. In the first case, we should first try to find a large building that has been talked about previously (given) and then enrich the assertion about the user's position with the id of this object. In the second case we can only know that a (new) large building is to the left and we do not know anything more about it. Usually, if a nominal phrase denotes something given, it will be stated in definite form and if it denotes something new it will be stated in indefinite form. The representation scheme needs to be enriched with this notion of the state of the information. The following examples show how this is done:

The large building is to my left <w:object xs:type="user">
 <w:position>
 <w:relation>
 <w:object xs:type="building">
 <s:extra>
 <s:info>given</s:info>
 </s:extra>
 <w:properties>
 <w:size>large</w:size>
 </w:properties>
 </w:object>
 <w:direction>250 290</w:direction>
 </w:relation>
 </w:position>
 </w:object>

A large building is to my left <w:object xs:type="user">
 <w:position>
 <w:relation>
 <w:object xs:type="building">
 <s:extra>
 <s:info>new</s:info>
 </s:extra>

```

        <w:properties>
          <w:size>large</w:size>
        </w:properties>
      </w:object>
    <w:direction>250 290</w:direction>
  </w:relation>
</w:position>
</w:object>

```

Notice that the extra information is put in the “semantics” namespace (“s”), since it lies outside the description of the world.

There are cases where a given object has not been explicitly introduced previously in the discourse. For example, one should expect that the utterance “I have come to a crossing” could be followed by “I pass **the crossing street** now”. The nominal phrase “the crossing street” is marked as given but the referent is not previously introduced explicitly. However, the mentioning of a crossing should also activate a set of other objects that can be treated as given in the following discourse. In this example, it would be possible to just add the referenced streets in the crossing object to the discourse history, and the crossing street would be found.

6.2 Speech acts

Since an underspecified object can be both a query and a statement, we need some representation of the speech act, i.e. the function of the utterance in the discourse. We will start by making the simple distinction between assertions and requests⁴. The difference between an assertion and a yes-no-request lies only in the speech act; the propositional content is the same:

The tree is large.

```

<s:assert>
  <w:object xs:type="tree">
    <s:extra><s:info>given</s:info></s:extra>
    <w:properties>
      <w:size>
        <s:extra><s:info>new</s:info></s:extra>
        large
      </w:size>
    </w:properties>
  </w:object>
</s:assert>

```

Is the tree large?

```

<s:request>
  <w:object xs:type="tree">
    <s:extra><s:info>given</s:info></s:extra>
    <w:properties>
      <w:size>
        <s:extra><s:info>new</s:info></s:extra>
        large
      </w:size>
    </w:properties>
  </w:object>
</s:request>

```

⁴ This is a highly simplified use the concept of “speech acts” (cf. Levinson, 1983, for a discussion). As long as we only consider obvious requests and assertions about the domain, this might suffice. The notion will probably be extended when we also consider other kinds of utterances.

In a wh-question, the question is not just if some description of the world is true, but some specific information is requested. The requested information is treated as a third information state similar to “given” and “new”.

What is the size of the tree?

```

<s:request>
  <w:object xs:type="tree">
    <s:extra><s:info>given</s:info></s:extra>
    <w:properties>
      <w:size>
        <s:extra><s:info>requested</s:info></s:extra>
      </w:size>
    </w:properties>
  </w:object>
</w:request>

```

7 Parsing

Now that we have defined what the semantic representation of utterances should look like, we need a parser that can build these representations from the user’s utterance. As we have seen, simple keyword spotting will not suffice for the domain. Instead, a parser using context-free grammars was chosen that is enhanced with some features for robust interpretation.

7.1 Parsing with context-free grammars

The basic idea in the parsing is to use an enhanced chart parser (cf. Jurafsky & Martin, 1997) with unification of typed XML structures. As an example, we will take the utterance “jag har en stor träbyggnad till höger” (“I have a large wooden building to my right”). The entries in the lexicon are shown below:

```

jag (i)      <g:rule cat="user" ag="subj">
              <g:match>jag</g:match>
              <g:sem><w:object xs:type="user"/></g:sem>
            </g:rule>
en (a)      <g:entry cat="art" numb="sing" gen="utr" info="new">
              <g:match>en</g:match>
              <g:sem><w:object/></g:sem>
            </g:entry>
stor (large) <g:entry cat="prop" declination="a1">
              <g:match>stor</g:match>
              <g:sem><w:size>big</w:size></g:sem>
            </g:entry>
träbyggnad <g:entry cat="lm" gen="utr" declination="n1">
(wooden    <g:match>träbyggnad</g:match>
building)  <g:sem>
              <w:object xs:type="building">
                <w:properties>
                  <w:material>wood</w:material>
                </w:properties>
              </w:object>
            </g:sem>
            </g:entry>

```

Each entry consists of a <match> part, which describes what the entry should match to, and a <sem> part, which describes the semantics that should be returned. The entry also has some specifications for grammatical category and linguistic features (i.e. number, given/new, gender)

In case of ambiguous words, these can just be repeated with different semantics in the lexicon. As can be seen, some entries (“stor” and “träbyggnad”) refer to declinations in a separate morphology (so that similar entries do not have to be repeated):

```

n1    <g:declination id="n1">
      <g:form numb="sing" info="new"/>
      <g:form numb="sing" info="given">en</g:form>
      <g:form numb="plur" info="new">er</g:form>
      <g:form numb="plur" info="given">erna</g:form>
    </g:declination>
a1    <g:declination id="a1">
      <g:form numb="sing" info="new" gen="utr"/>
      <g:form numb="sing" info="new" gen="neutr">t</g:form>
      <g:form numb="sing" info="given">a</g:form>
      <g:form numb="plur">a</g:form>
    </g:declination>

```

The rules in the grammar look like this:

```

“en stor          <g:rule cat="lmp">
träbyggnad”      <g:match congruent="numb info gen">
(“a large wooden <g:entry cat="art"/>
building”)        <g:entry cat="prop" link="1"/>
                  <g:entry cat="lm" link="2"/>
                  </g:match>
                  <g:sem>
                    <g:unify copy="info numb">
                      <g:ref link="2"/>
                      <g:ref link="1"/>
                    </g:unify>
                  </g:sem>
                </g:rule>

```

The matching part of the rule is very flexible and can match to anything that is given in an entry or a rule, but also to specific words (which means that the parser can work without a lexicon). They can also include expressions such as “one-of” or “optional” which makes the matching more robust and the rules more general. In this example, the matching tag says that number and info should be congruent in the entries. The resulting congruence nodes (number and info) in the interpretation are implicitly copied to the rule. Thus, they are not stated explicitly in the rule, since they are dependent on the matching entries. In this case, the rule says that the resulting semantics should be the result of unifying the semantics of the matching parts. The “copy” instruction says that the parser should also put the “info” part of the resulting congruence in an “extra” node directly under the first semantic element in the entries. The reason for this copy-method is that the semantics of given and new cannot be stated directly in the semantics of the entries, since they could be non-congruent (see section 8.3).

7.2 Semantic unification

The unification in this parser is somewhat different from the traditional unification algorithm (cf. Jurafsky & Martin, 1997) in that it does not require the unifying structures to start at the same level. In this example, it will unify `<w:size>big</w:size>` with

```

<w:object xs:type="building">
  <w:properties>
    <w:material>wood</w:material>
  </w:properties>
</w:object>

```

The parser can use knowledge about the domain structure (represented with the XML Schema of the domain XML) to find out that the “size” element fits under the “properties” element under “object” and the result will be:

```

<w:object xs:type="building">
  <w:properties>

```

```

    <w:material>wood</w:material>
    <w:size>large</w:size>
  </w:properties>
</w:object>

```

This differs from traditional unification or the use of lambda expressions (cf. Jurafsky & Martin, 1997), where this unification would fail or the structure would be put in the wrong place. The advantage is that we do not need to know exactly what the semantics of the matching parts actually looks like. In for example Strömbäck & Jönsson (1998), the unifying paths must be stated explicitly in the grammar rules. In this parser, we can make the rules more general and robust. The semantics of the individual words together with the system's model of the world are given precedence over the syntactic structure in the resulting semantic representation.

The resulting semantics of the nominal phrase after applying the rule will look like this:

```

<w:object xs:type="building">
  <s:extra><s:info>given</s:info></s:extra>
  <w:properties>
    <w:material>wood</w:material>
    <w:size>
      <s:extra><s:info>given</s:info></s:extra>
      large
    </w:size>
  </w:properties>
</w:object>

```

We now need to add the rule for the relation and the whole assertion:

```

“till höger”      <g:rule cat="rel">
(“to my right”)  <g:match>
                  <g:optional>till</g:optional>
                  höger
                  <g:optional>om</g:optional>
                </g:match>
                <g:sem>
                  <w:relation>
                    <w:direction>80 100</w:direction>
                  </w:relation>
                </g:sem>
              </g:rule>
“jag har en stor  <g:rule cat="s">
träbyggnad till  <g:match>
höger”           <g:entry cat="user" ag="subj" link="1"/>
                 <g:optional>har</g:optional>
                 <g:rule cat="lmprel" link="2"/>
                 <g:optional>
                   och
                 <g:rule cat="lmprel" link="3"/>
                 </g:optional>
               </g:match>
               <g:sem>
                 <s:assert>
                   <g:unify>
                     <g:unify/>
                     <w:distance>0 500</w:distance>
                   </g:unify>
                 </s:assert>
               </g:sem>
             </g:rule>

```

The resulting semantics will look like this:

```

<s:assert>

```

```

<w:object xs:type="user">
  <w:position>
    <w:relation>
      <w:direction>260 280</w:direction>
      <w:distance>0 500</w:distance>
      <w:object xs:type="building">
        <w:properties>
          <w:size>
            big
            <s:extra>
              <s:numb>sing</s:numb>
              <s:info>new</s:info>
            </s:extra>
          </w:size>
          <w:material>wood</w:material>
        </w:properties>
        <s:extra>
          <s:numb>sing</s:numb>
          <s:info>new</s:info>
        </s:extra>
      </w:object>
    </w:relation>
  </w:position>
</w:object>
</s:assert>

```

The parser also returns the resulting semantics along with the parse tree. The XML now starts to be hard to read (it is not shown here), so we apply an XSLT style sheet to convert it into HTML and display it in a web browser, so that it is easier to read. The output is shown in Figure 3.

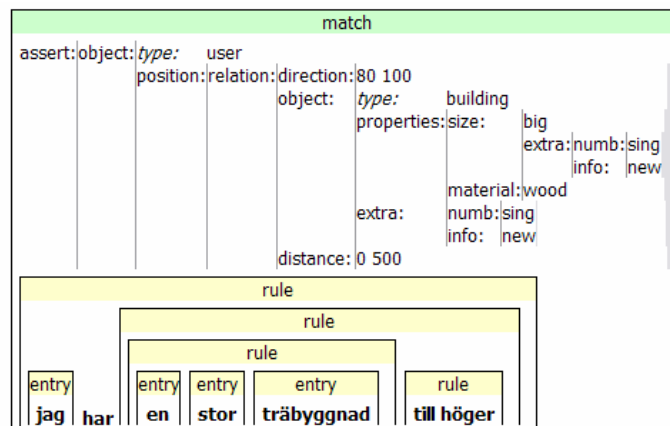


Figure 3. The parse result after applying an XSLT style sheet that converts it into HTML (“I have a large wooden building to my right”).

8 Robust interpretation

The parser will receive a set of hypotheses from the speech recogniser. Since we want to use n-gram grammars in the speech recognition, we cannot expect the input to be “grammatically correct”. However, the parser should come up with an interpretation that is as good as possible anyway.

8.1 Best combination of phrases

If the parser does not find complete speech acts, it should return the best solution anyway. This is shown in Figure 4. The solution that consists of the fewest number of rules, while

covering the most number of words is chosen. The solutions are found by searching the chart for the best combinations of passive edges. The words that are not part of the solution are marked with “nomatch”. If there are different solutions which are not semantically equivalent, they are returned in an n-best list which is based on coverage and parsing confidence score (see section 8.4).

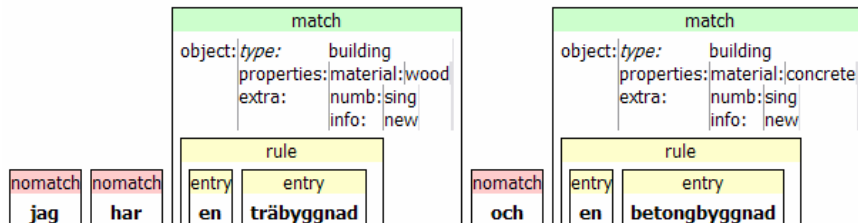


Figure 4. A parse that does not cover the whole utterance (“I have a wooden building and a concrete building”). There are two matches with separate parse trees in the result.

8.2 Insertions

Another feature of the parser is that it can handle insertions in the input. Insertions can appear in any rule anywhere due to disfluences such as “eh” and truncated words, resulting in (seemingly) random words. Instead of putting in optional insertions in all rules, the parser will automatically detect and ignore them. This is done by letting the active edges in the chart accept non-expected words and continue to look at the next word.

The total parsing score (see section 8.4) will be lowered by such insertions (depending on the confidence of the inserted word) and only a limited number of subsequent insertions are allowed (which is also due to computational limitations). An example of insertions in the input is shown in Figure 5. Notice that many of the inserted words actually are in-vocabulary (such as “tree”, “left”, “a”), and that the parser still classifies these as insertions.

8.3 Non-congruence

Congruence errors can be a problem in speech recognition, since morphological inflections may disappear or be introduced. This is often not a problem for keyword spotting (where morphological inflections are just ignored), but if we want to make semantic differences between morphological forms, as in the HIGGINS domain, it may be. Consider the phrase “den röd byggnad”, which is acoustically similar to “en röd byggnad” or “den röda byggnaden”. Should the building be considered as given or new? As we have seen, the matching part of the rule states that all parts should be congruent:

```
<g:match congruent="numb info gen">
  <g:entry cat="art"/>
  <g:entry cat="prop" link="1"/>
  <g:entry cat="lm" link="2"/>
</g:match>
```

However, the parser will also accept non-congruent sequences, such as “den röd byggnad”. The semantics part of the rules states that “info” and “numb” should be copied to the resulting node:

```
<g:sem>
  <g:unify copy="info numb">
    <g:ref link="2"/>
    <g:ref link="1"/>
  </g:unify>
</g:sem>
```

Since the parts are non-congruent, the node will copy the features that are most probable, given the speech recogniser’s word confidence scores, and result in several interpretations. A

non-congruent phrase should of course result in a lower parsing score (see section 8.4). An example of a parse of a non-congruent phrase (“den stor träbyggnad”) is shown in Figure 5.

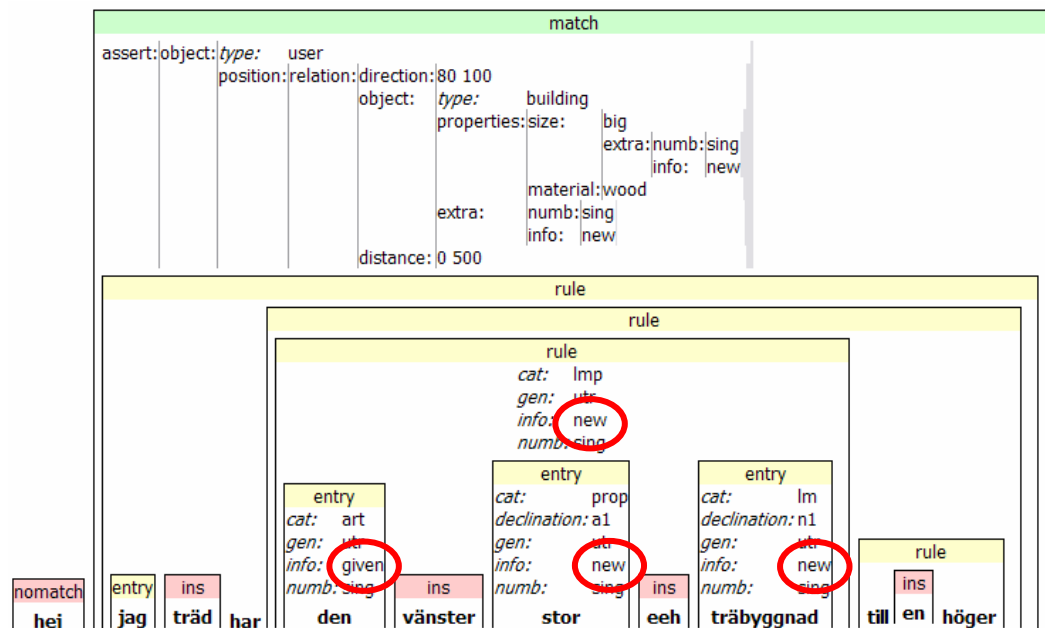


Figure 5: A parse result of a “non-grammatical” noisy input (“hello I tree have the left large eeh wooden building to a right”). The resulting semantics is exactly the same as in Figure 3. The circles mark how the non-congruence is handled and solved at the higher node in the parse tree.

8.4 Parsing confidence score

To guide the interpretation, the speech recogniser also provides word confidence scores that should be used by the parser. The parser should also provide parsing confidence scores along with the interpretations. This parsing confidence score should be based on the word confidence scores, but also on insertions and non-congruence. The insertion of a word with high confidence should lower the confidence of the phrase more than the insertion of a word with low confidence. How the parsing confidence score should be computed must be tested against real speech recognition data, and is a topic for future research.

The parsing confidence score should be used in the dialogue management in order to take the right grounding actions. An interpretation that has a low confidence should perhaps be verified against the user, whereas an interpretation with a high confidence can be accepted by the system without confirmation. Other factors than the parsing score should of course also be considered when selecting the amount of feedback that the system should give, such as the consequence of task failure when accepting an interpretation.

8.5 Incremental parsing and self-corrections

The parser also supports incremental parsing. Thus, each time the speech recogniser returns a new word (while the user is speaking), the word is added to the chart, new passive edges are built, and the chart is searched for best combinations of passive edges.

Incremental parsing could be used for several things. First, other components than the speech recogniser could determine that the user has finished speaking and it is the system’s turn. This is normally decided based on the amount of silence from the user, which can re-

sult in pauses between each turn. If the turn taking instead could be based on semantics, it could perhaps speed up turn taking and the system could even barge-in if it was appropriate.

The system could also possibly elicit feedback while the user is speaking. This has not been tested so far with real users or speech recognition data, but it is technically possible. Imagine the following dialogue, where a “mumbling” text synthesis is playing back the semantics that it has got so far:

User:	Jag har ett träd ...	(I have a tree ...)
System:	en träbyggnad	(a wooden building)
User:	... nej, ett träd ...	(... no, a tree ...)
System:	ett träd	(a tree)
User:	... till vänster	(... to my left)
System:	du har ett träd till vänster	you have a tree to your left

The user must in such a case, of course, also be able to make self-corrections in the same parse. This can easily be achieved by adding the following rule:

```
<g:rule cat="lmp">
  <g:match>
    <g:rule cat="lmp"/>
    nej
    <g:rule cat="lmp" link="1"/>
  </g:match>
  <g:sem>
    <g:ref link="1"/>
  </g:sem>
</g:rule>
```

The rules states that a landmark phrase (“lmp”) could consist of two landmark phrases with a correction word between (“no”). The semantics part of the rule says that only the semantics of the second landmark phrase is returned. The resulting parse is shown in Figure 6. One cannot, of course, always rely on a correction word between the landmark phrases, especially since such as short word can easily be lost in the speech recognition. Two landmark phrases could also form a conjunction or, less probably, a disjunction. In future research, we will try to use prosodic information to make this distinction.

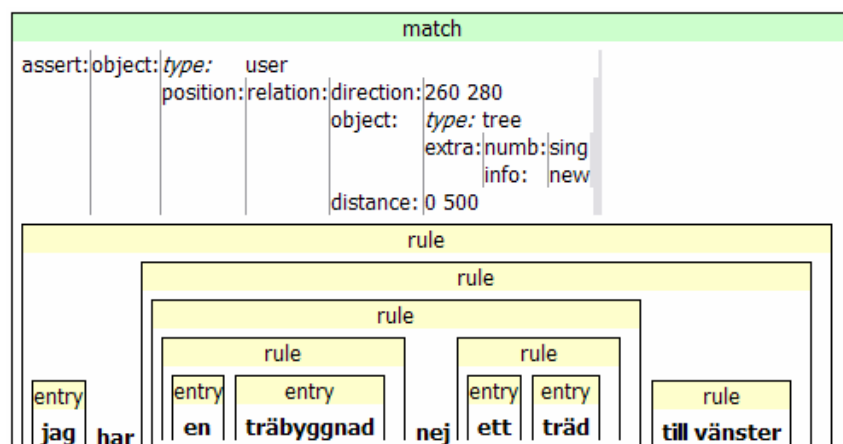


Figure 6: A parse result with a self-correction (“I have a wooden building no a tree to my left”). Notice that the semantics does not contain anything about the corrected phrase “a wooden building”.

9 Discussion

This paper has described how noisy results from the speech recogniser can be robustly parsed and interpreted for use in a spoken dialogue system. However, this technical report needs to be followed up with tests on real speech recognition data in order to verify the feasibility of the techniques.

Only the representations of utterances that are about the domain have been discussed in this paper. The dialogue system should of course also account for utterances that are concerned with the interaction, such as acknowledgements, grounding, request for repetitions, etc.

The semantic object model and parsing techniques presented here should be applicable to other domains as well, including information seeking, since it supports talking about the properties and relations of objects in general. It should be fairly straightforward to use it for more traditional domains, such as travel information.

The subject-verb-object order of the semantic model also has the same structure as RDF – the XML standard for the semantic web (RDF Core Working Group, 1999). Thus, it would be interesting to study to what extent it is possible to convert these semantic descriptions to RDF.

10 References

- Bell, L., Boye, J., & Gustafson, J. (2001). Real-time handling of fragmented utterances. *Proceedings of the NAACL 2001 Workshop on Adaptation in Dialogue Systems*, Pittsburg, USA.
- Brown, G., & Yule, G. (1983). *Discourse Analysis*. Cambridge University Press, Cambridge.
- Flycht-Eriksson, A. (2001). *Domain Knowledge Management in Information-providing Dialogue Systems*. Licentiate Thesis, Linköping Studies in Science and Technology, Thesis No. 890, School of Engineering at Linköping University ISBN: 91-7373-050-5
- Gustafson, J., Bell, L., Beskow, J., Boye, J., Carlson, R., Edlund, J., Granström, B., House, D. & Wirén, M. (2000). AdApt - a multimodal onversational dialogue system in an apartment domain. *Proceedings of ICSLP 2000*, 2:134-137.
- Heisterkamp, P., McGlashan, S., & Youd, N. (1992). Dialogue Semantics for an Oral Dialogue System. *Proceedings of the International Conference of Spoken Language Processing*, Banff, Canada.
- Jurafsky, D. S., & J. H. Martin. (2000). *Speech and Language Processing*. Prentice Hall, Inc., Englewood, N.J.
- Jönsson, A. (1997). A model for habitable and efficient dialogue management for natural language interaction, *Natural Language Engineering* 3(2/3), pp 103-122.
- Larsson, S. (2002). *Issue-based Dialogue Management*. PhD Thesis, Göteborg University.
- Levinson, S. C. (1983). *Pragmatics*. Cambridge University Press, Cambridge.
- Melin H. (2001). ATLAS: A generic software platform for speech technology based applications. *TMH-QPSR*, KTH, Stockholm, 1:29-42.
- Poesio, M. (1996). Formal Semantics and Spoken Dialogues. *Proceedings of the ECAI Workshop on Corpus-Based Semantic Analysis*.
- RDF Core Working Group. (1999). Resource Description Framework (RDF) Model and Syntax Specification. Available: <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/>

- Skantze, G. (2003). Exploring Human Error Handling Strategies: Implications for Spoken Dialogue Systems. *Proceedings of the ISCA Tutorial and Research Workshop on Error Handling in Spoken Dialogue Systems*.
- Strömbäck, L., & Jönsson, A. (1998). Robust Interpretation for Spoken Dialogue Systems, *Proceedings of ICSLP'98*, Sydney, Australia, pp. 491-495.
- XML Core Working Group. (2000). Extensible Markup Language (XML) 1.0 (Second Edition). Available: <http://www.w3.org/TR/REC-xml>.
- XML Schema Working Group. (2001). XML Schema specification. Available: <http://www.w3.org/TR/xmlschema-0/>; <http://www.w3.org/TR/xmlschema-1/>; <http://www.w3.org/TR/xmlschema-2/>
- XSL Working Group. (1999a). XML Path Language (XPath) Version 1.0. Available: <http://www.w3.org/TR/xpath>
- XSL Working Group. (1999b). XSL Transformations (XSLT) Version 1.0. Available: <http://www.w3.org/TR/xslt>