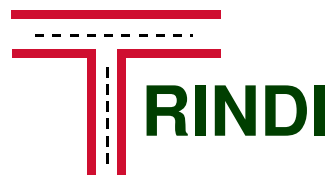

Robust interpretation and dialogue dynamics

Annie Zaenen Stina Ericsson Staffan Larsson
A. Mikheev David Milward Manfred Pinkal
Massimo Poesio CJ Rupp Karsten Worm

Distribution: PUBLIC



Task Oriented Instructional Dialogue
LE4-8314

Deliverable D5.2

May 2000

Task Oriented Instructional Dialogue

Gothenburg University

Department of Linguistics

University of Edinburgh

Centre for Cognitive Science and Language Technology Group, Human Communication Research Centre

Universität des Saarlandes

Department of Computational Linguistics

SRI Cambridge

Xerox Research Centre Europe

For copies of reports, updates on project activities and other TRINDI-related information, contact:

The TRINDI Project Administrator
Department of Linguistics
Göteborg University
Box 200
S-405 30 Gothenburg, Sweden
trindi@ling.gu.se

Copies of reports and other material can also be accessed from the project's homepage, <http://www.ling.gu.se/research/projects/trindi>.

©2000, The Individual Authors

No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission from the copyright owner.

Chapter 2 was written by David Milward, Chapter 3 by Manfred Pinkal, CJ Rupp and Karsten Worm, Chapter 4 by Massimo Poesio and A. Mikheev and Chapter 5 by Stina Ericsson and Staffan Larsson. The deliverable was edited by Annie Zaenen.

Contents

1	Introduction	9
2	Robust Semantics for Dialogue: Flat Structures	10
2.1	Introduction	10
2.2	Introduction to Flat Structures	11
2.3	Dialogue Systems	14
2.3.1	Current Systems	14
2.3.2	Requirements for Dialogue Systems	15
2.3.3	Summary	20
2.4	An Approach using Flat Structures	20
2.4.1	Choice of Flat Structure	20
2.4.2	Building a Flat Structure	21
2.4.3	Applying rules to a semantic chart	22
2.5	Evaluation of the Flat Structure Approach	27
2.6	A preliminary implementation	29
2.7	Conclusions	30

3	Robust Semantic Processing of Spoken Language: Combining Fragments	31
3.1	Introduction	31
3.2	Robustness	32
3.2.1	Robustness and Graceful Degradation	32
3.2.2	Robust Processing of Spoken Language	32
3.2.3	Robust Semantics Processing	33
3.3	Architecture	34
3.3.1	Accurate and Robust Parsing of Word Lattices	34
3.3.2	Efficiency and Time Constraints	35
3.3.3	Multiparser Architecture	36
3.3.4	The Various Parsers	37
3.4	The Description of Robustness Operations	37
3.4.1	The Relevant Levels of Information	38
3.4.2	Semantic Representations: The VIT Format	39
3.4.3	The Rule Syntax	39
3.4.4	Confidence Values for Operations	42
3.5	The Processing Model	42
3.5.1	Storing Partial Analyses	43
3.5.2	Combining Partial Analyses	44
3.5.3	Scoring and Result Selection	46
3.6	The Rule Sets	49
3.6.1	Rule Classification and Examples	49

3.7	Evaluation	55
3.8	Conclusion	56
4	A Statistical Model for Predicting Dialogue Moves on the Basis of Game Structure	57
4.1	Introduction	57
4.2	Game-Based Expectations	58
4.3	Methods	59
4.4	Experiments	60
4.4.1	Baseline	60
4.4.2	First Experiment - Bigrams	60
4.4.3	Second Experiment - Information about Games	60
4.4.4	Third Experiment: Tracking Speaker Change	61
4.4.5	Fourth Experiment - A more complex notion of game structure	61
4.5	Related Work	61
4.6	Conclusions	62
5	Finite state approaches to dialogue dynamics	63
5.1	From Utterance to Dialogue Move	63
5.1.1	Synonymy sets, Concepts, and Move Grammar	64
5.2	Updating information states	67
5.2.1	Operations and conditions	67
5.2.2	Information state update rules	69

5.2.3	Generating FSTs from TRINDiKIT update rules	70
5.2.4	Using flag diacritics	70
5.3	Conclusions and further research	71

Chapter 1

Introduction

A central problem with the implementation of the interpretation of natural language utterances, despite the current effort on underspecified interpretation, is that the classical approach of computing some kind of semantic representation on the basis of parsing an utterance leads to brittle systems which are unusable on free natural language input. In this work package we will attempt to characterise more generally in what way more robust techniques can be applied to or combined with semantic processing. It is possible to apply techniques related to shallow parsing to the recognition of linguistic aspects of dialogue moves, i.e. it need not be necessary to give a complete analysis of a dialogue contribution in order to recognize what kind of dialogue game is involved. It is an open question to what extent an information state could be partially characterized on the basis of a “light analysis” of a dialogue contribution. For example, the incorporation of lexical semantic information into constraint grammars may possibly provide more robust ways of ambiguity resolution than are currently available for natural language interpretation.

The SRI contribution looks at how the interpretation of parts of an utterance can be affected by the context, including other parts of the utterance. It also describes how flat or ‘distributed’ semantic representations can be used to give both the robustness you might expect from a shallow analysis, combined with the ability to respect constituency and scope (where it can be determined) that you might expect from a deep analysis. The Saarbruecken contribution elaborates on the Verbmobil experience. The Edinburgh contribution discusses a method for building a statistical model for predicting dialogue moves on the basis of an annotated corpus. The Gothenburg contribution describes an approach to shallow parsing using finite-state techniques for dialogue move recognition, and explores the possibility of using finite state technology for updating information states.

Chapter 2

Robust Semantics for Dialogue: Flat Structures

Dialogue utterances are rarely sentences and are often fragmentary^{1 2}. This paper discusses some of the implicit assumptions made by shallow and deep processing approaches, and advocates a new approach which keeps the robustness of shallow or keyword-based approaches, whilst retaining the generality and formal nature of a full semantics.

2.1 Introduction

The ideal spoken dialogue system should be flexible, allowing users to supply extra information from that specifically asked for, or to take the initiative. However, these aims can be difficult to square with the need for top down expectation to help speech recognition accuracy. Thus there tends to be a divide between tightly constrained systems with e.g. separate finite state grammars to recognise users responses to particular system queries, and unconstrained systems which rely on keyword spotting/pattern matching, or attempt deep language analysis. In this paper we will try to bridge various divides between shallow and deep processing systems and show how top down expectation from dialogue context can still be incorporated in a flexible system. The key to the approach is to use a ‘flat’ representation where information about the semantic (or syntactic) structure is distrib-

¹This chapter appears as Milward, D. ”Towards a Robust Semantics for Dialogue using Flat Structures”, Proceedings of Amstelogue ’99 Part II

²I would like to thank other members of the Trindi Consortium for their comments on this work, in particular Robin Cooper, Ian Lewin, Manfred Pinkal and Massimo Poesio for various discussions which fed into the work. I would also like to thank Ian Lewin and Rob Koeling for their comments on an earlier draft.

uted between a set of constraints. This makes it possible to combine the robustness we would expect from pattern matching without wasting any available linguistic information concerning constituent structure.

The paper is split into four main parts. Section 2 describes flat structures. Section 3 describes various requirements and issues in dialogue processing. Section 4 describes an approach to dialogue processing using flat structures. Section 5 evaluates the approach according to the requirements described in Section 3. Section 6 describes a preliminary implementation.

2.2 Introduction to Flat Structures

The most basic form of flat structure is just an indexed version of a standard semantic representation. Consider a standard recursively structured piece of semantic representation:

$P(a, Q(b, c))$

This can be represented as the application structure:

```

      .(.,.)
     / | \
    P a  .(.,.)
         / | \
        Q b  c
  
```

Now consider giving each item in the structure a unique label i.e.

```

    i1: .(.,.)
       / | \
    i2:P i3:a i4: .(.,.)
           / | \
          i5:Q i6:b i7:c
  
```

This information can be represented by the set of constraints:

$i1:i2(i3,i4), i2:P, i3:a, i4:i5(i6,i7), i5:Q, i6:b, i7:c$

The resulting representation is flat in the sense that there is no explicit recursion. The set of constraints, $i2:P$, $i3:a$, $i5:Q$, $i6:b$, $i7:c$ describe the lexical/morphological content of the representation. The constraints $i1:i2(i3,i4)$ and $i4:i5(i6,i7)$ describe the structural aspects. For example, consider the representation we might get for the sentence:

John believes Jack runs

The following flat structure corresponds to the logical form, $believe(john,run(jack))$:

$i1:i2(i3,i4)$, $i2:believe$, $i3:john$, $i4:i5(i6)$, $i5:run$, $i6:jack$

This provides the morphemes, $believe$, $john$, run and $jack$ along with constraints specifying their relationship.

Note that we have only changed how we represent the semantics: there is a one-to-one mapping between the set of constraints and the original recursive representation (assuming index renumbering). The basic flat structure can be regarded as an alternative notation, or as a description of the original semantic representation.

It should be noted that there are many different varieties of indexed/flat structures, going back at least to Kay (1970). For example, neo-Davidsonian semantics is sometimes described as a flat representation, since event variables act somewhat similarly to indices. The semantics for a sentence such as "John runs at 5pm" is given by a conjunction of two constraints hanging off an event variable i.e.

$\exists run(e,j) \ \& \ at(e,5)$

This enables inferences such as "John runs at 5pm" therefore "John runs" to go through without the need for meaning postulates. Hobbs (1983) extended this approach to all predicates by mapping each n-ary predicate to an n+1-ary predicate including an event variable, thus allowing restrictive modification to be done via conjunctive constraints.

A rather different kind of flat structure (closer to the basic flat structures described above) has been used as a way to provide semantics for fragments which do not form standard constituents (Milward 1991). For example, the semantics of "Mary Fred" in the sentence "John showed Mary Fred or Peter Sue" is treated as the set of constraints, $i4:mary$, $i5:fred$. An intermediate semantic representation (prior to quantifier and conjunction scoping) uses conjunction or disjunction of sets of constraints e.g. $OR(i4:mary, i5:fred, i4:peter, i5:sue)$.

More recently, flat structures which combine both a Davidsonian approach to modification with indexing similar to basic flat structures have been used in Machine Translation (Copestake et al. 1995) following the lexicalist approach of Whitelock (1992) and Trujillo (1995). The prime motivation within Machine Translation is that we can more easily express translation rules which pick up disconnected pieces of the source semantic representation and map them to a single piece of the target representation (or vice versa). Transfer between source language and target language representations is achieved by mapping between a subset of the conjuncts in the source to another set of conjuncts in the target. The translation process is inherently bottom up: we translate subset by subset, and the final set of target conjuncts should comprise a fully connected semantic representation.

Finally, there has also been much interest in using indexed representations for underspecified semantic representation (e.g. Reyle 1993, Egg 1998). Here the emphasis has mainly been on weakening structural constraints to enable underspecified representation of quantifier scope. Structural constraints are divided into dominance, precedence and immediate dominance constraints (similar to the work of Marcus et al. 1983 on the description of syntactic tree structure) making it possible to state that a piece of representation is within the scope of another, without further specifying the relationship.

The interest in flat structures here is motivated by the distributed nature of the representation. There is thus a better chance of achieving robustness when dealing with fragmentary input, or in making rules (e.g. mapping to database slot-values) sensitive to other parts of a representation. To illustrate the former, consider a case where a structural analysis has produced three separate fragments P, a, and Q(b,c). This information can be represented by the set of constraints:

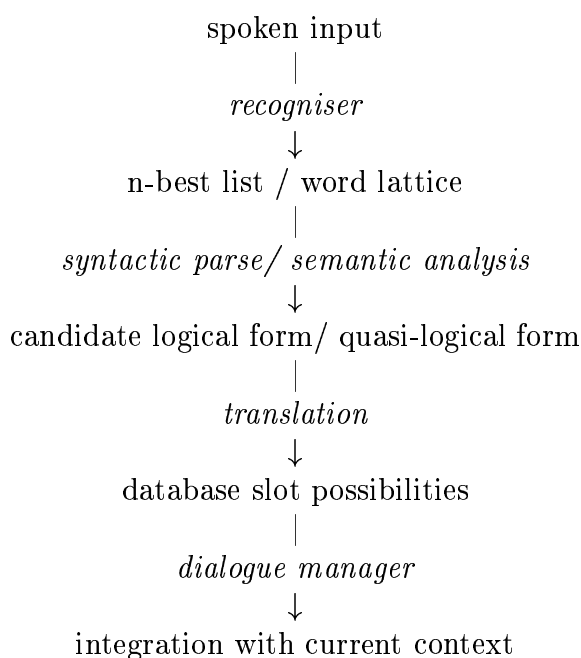
i2:P, i3:a, i4:i5(i6,i7), i5:Q, i6:b, i7:c

This is a subset of the flat structured representations for P(a,Q(b,c)), with only the constraint, i1:i2(i3,i4) missing. The change of representations thus has the advantage of making the semantics of fragments and full constituents look more similar. We will see that this in turn suggests ways in which rules (e.g. for mapping to database slot-values) can be relaxed to ensure robustness.

2.3 Dialogue Systems

2.3.1 Current Systems

Flexible dialogue systems tend to be either based on shallow processing (e.g. keyword spotting or pattern matching) or deep processing (interpretation down to a level of logical or quasi logical form). Shallow systems are task directed, and do not attempt a full semantic or pragmatic analysis of all parts of an utterance, although processing may be reasonably detailed for the parts relevant to the task. Deep processing systems tend to have a layered architecture e.g.



This is the basis of the architecture of the deep processing component of the SLT system (Boye et al., 1999). The SLT system keeps the top 5 analyses produced by the recogniser to allow later stages of processing to act as a filter on the results of the recogniser. Statistically trained triples (Carter 1997) build in domain dependence to choose the best syntactic/semantic parse. Other systems such as Verbmobil (Kasper et al. 1999, Goertz et al. 1999) and OVIS (van Noord et al, 1999) keep the recogniser lattice, annotating this with syntactic and semantic information.

There is often an assumption that deep approaches will provide better results than shallow approaches in the long term. All we need is more work on our grammars and faster machines. In the meantime we should use multi-engined approaches where you use full

analysis wherever possible, but back off to shallow analysis where deep analysis fails (either through time out or lack of coverage). This kind of argument was very much the inspiration for the SLT system and it assumes that when deep analysis provides a result it will generally be better than the result of shallow processing. However, empirical evaluation of the resulting system gave exactly the opposite result. Lewin et. al. (1999) report that when the results of the two analysis modes disagreed, the shallow analysis provided a better result three times as often. One of the inspirations for this paper was to try to see if there might be some theoretical reasons for these results, so let us now consider how various issues tend to be handled by the two kinds of systems.

2.3.2 Requirements for Dialogue Systems

Robustness

Shallow and deep systems tend to tackle the issue of robustness differently. Many shallow approaches home in on words or phrases which are relevant to the task. This means that unknown vocabulary or unknown grammatical constructions will normally have little or no affect on the processing. To give an example, a keyword based approach which only looks for city names should easily deal with an interchange such as:

Where do you want to go?

Well let me see now, um, well I think Boston, yes that's right

In contrast, a deep approach will normally come up with a set of possibly overlapping analysed fragments. There are then two further tasks. The first is to pick the best set of fragments. The second is how to translate the fragments into some kind of database update function (or whatever is required for the task). A common approach to the first task (e.g. Verbmobil) is to prefer the shortest possible path through a set of fragments i.e. the smallest number of fragments which span the utterance. When a single constituent spans the whole utterance this will be picked since the path is just of length 1. No contextual information is included at this stage. For the second task, there are two common approaches. The first is to translate the fragments independently into database update functions. The second is to apply various reconstruction rules (working on either syntactic or semantic fragments) to try to fit the pieces into a full sentence (c.f. Verbmobil).

Domain Dependence

Consider the following sentence in the Air Traffic Domain:

Show flights to Boston

This has two readings, one where “flights to Boston” is a constituent, the other where “to Boston” is an adverbial modifier (similar to “to Fred” in “Show flights to Fred”).

A shallow system may well have rules for “flights to <city>”, but is unlikely to include rules for the adverbial reading. Thus the possibility of ambiguity does not exist. Despite there being no explicit domain training, the correct reading is picked.

In contrast, in a deep analysis system with a general purpose grammar, there has to be specific customisation to the domain. This may be via specialisation of the grammar to the domain (c.f. OVIS), or via the introduction of domain specific preference mechanisms. For example, the SLT system uses ‘treebanking’ (Carter 1997) which involves a human picking correct analyses in order for the machine to learn domain specific syntactic and semantic triples which select between alternative readings.

Other examples may be problematic for shallow systems which employ domain independent chunking. For example, consider the following interchange:

Please give me your departure and destination cities

Boston London Heathrow

The correct bracketing here requires the use of domain specific information that “London Heathrow” can be treated as a unit. To use this information we either need to incorporate a domain specific pre-processing stage which picks out idiomatic expressions, or preserve alternative bracketings until later domain specific processing.

Finally we should note that domain dependence (and context dependence as we shall discuss below) should not just be used for choosing between readings given by full parses of the utterance, but should affect which fragments are picked. Sometimes this will mean that a fragment interpretation should be chosen instead of a full interpretation. For example, a relatively common problem with the current SLT system is where the recogniser suggests an extra word at the end of an utterance. A full analysis can often be found which incorrectly incorporates the bogus word, and this is picked in preference to a correct fragment analysis.

Context Dependence: Previous Utterance

Shallow systems typically incorporate preferences arising from the preceding question. This is used to ensure that the answer “Boston” in the context of the question “Where do you want to go” is interpreted differently from in the context of “Where do you want to leave from”. In deep systems the need for context dependent preferences is not so immediately obvious, since examples such as these can be treated using some variety of ellipsis resolution where the utterance is first translated into something equivalent to “I want to go to Boston”. This however breaks down in cases where there is a full sentence reply. Consider the following example (which occurred in the SLT system). The interchange is as follows:

S: Where would you like to go?

U: I would like to go to/from Boston

The speech recogniser fails to distinguish between to/from, and domain specific preferences happen to very slightly prefer “from” vs . “to” in this context. The system thus decides (incorrectly) that the most likely sentence is “I would like to go from Boston”, The correct analysis has now been ruled out, and the dialogue manager cannot recover the correct interpretation.

How can this be changed? There are two options. The first is to bring contextual information to bear much earlier in the process (this is effectively what is happening in the shallow approaches). The second is to ensure all analyses survive until a late stage, where context then comes into play. In the SLT system domain specific preferences could be context specific as well, though this would entail a larger treebanking stage. In OVIS there has already been some experimentation with bringing in context early in the process and weighting fragments accordingly.

Context Dependence: Other parts of the utterance

Consider the following examples:

I'd like to leave York, now let's see, yes, at 3pm

at 3pm \Rightarrow departure_time(3pm)

I'd like to arrive at York, now let's see, yes, at 3 pm

at 3pm \Rightarrow arrival_time(3pm)

The translation of the phrase “at 3pm” is dependent here not on any outside context but on the rest of the utterance. This is relatively easy to cope with in a shallow processing system which works on a single analysis at a time. We merely need patterns of the form:

[leave] [at 3pm]/time(3pm) \Rightarrow departure_time = 3pm
[arrive] [at 3pm]/time(3pm) \Rightarrow arrival_time = 3pm

There is no a priori reason to suggest that we could not apply similar context dependent rules within a deep approach. although it might be necessary to apply incomplete heuristics to avoid the inefficiency caused by fragments not being able to be translated independently.

Reconfigurability

It is sometimes argued that deep approaches are more easily reconfigurable since there is more reuse of standard processing modules (e.g. parsing, morphology and semantic construction). Shallow processing systems tend to be more task directed, so we might expect a greater proportion of the code to need changing. However in practice this argument only tends to apply when there is a change of task (e.g. enquiries regarding routes vs. prices), but no change of domain (e.g. computer manuals vs. airline booking). When moving to a new domain a deep approach has more work to do since it doesn't just have to deal with words or phrases of particular interest to the task, but all new words and constructions found in the domain.

Where shallow systems tend to be more problematic is when we try to improve coverage. Many systems rely on precisely ordered pattern matching rules. Adding a rule or changing the order for one phenomenon often causes another to break. This is not unlike trying to maintain a large set of grammar rules in a deep analysis system.

Accuracy

The final issue to consider is the accuracy of analyses produced by shallow or deep systems. A feature of most shallow systems is that they are goal directed. Similar to Information Extraction (IE) Systems, they only look for the information they need and ignore the rest. Let us look at an example from IE first, where the issues are particularly clear. A standard task in IE is to look for relationships between people and companies e.g. “who is chairman of which company”. Consider the following sentence:

John Smith, Director of ABC Corp., announced today that his four task force

managers had increased profits in three divisions.

From this we can infer that John Smith is Director of ABC Corp. without unpacking all the readings for the sentence as a whole. This is a valid inference to make since the sentence can be rephrased as the following conjunction.

John Smith is Director of ABC Corp. and John Smith announced ...

It is safe to infer “A” from “A and B” even if “B” is ambiguous.

Another standard abstraction made in IE is to ignore adjectives or other noun modifiers and just look at the head noun. For example, a system might extract “ABC Corp. made a profit” from all three sentences below, although it is only safe to do so from the first:

ABC Corp. announced a pre-tax profit of \$4000 dollars
ABC Corp. falsely announced a pre-tax profit of \$4000 dollars
ABC Corp. announced a false pre-tax profit of \$4000 dollars

Dialogue is a bit different since we don’t usually have to wade through immense quantities of irrelevant text. However, there are plenty of cases where information is supplied which there is no need to process. For example in the flight booking domain we get cases such as:

I would like a comfortable flight to Boston because my sister

Here again we can ignore the reason, and the modifier, “comfortable”, to extract the relevant request of “I would like a flight to Boston”.

There are also similar cases in dialogue where it is not safe to ignore surrounding material e.g. it is safe to use shallow processing to pick out “destination = Boston” in the first example below, but not the second

I would like to go to Boston
Now let me see, not to Boston, but perhaps to Chicago

How can we deal with this kind of negative information? The first option is just to ignore it. Negation (implicit or explicit) and non-intersective modification is not that common in

newspaper texts or in many dialogue scenarios, so the correct inferences will normally go through. However, this means we are unnecessarily throwing away useful information. The second option is to perform some checking of the monotonicity properties of the context e.g. by checking there is no ‘not’ with the potential to have scope over the material of interest. The third option is to move closer to deep processing, since part of the job of full parsing is to determine constituency, hence scoping, and the job of a grammar to provide useful generalisations using the recursive structure of a language.

2.3.3 Summary

What conclusions can we make from the discussion above? Firstly, the shallow processing paradigm does have theoretical justification. In shallow processing you make (defeasible) inferences using partial information about the utterance content. This is similar to how Oaksford and Chater (1991) argue that humans generally have to behave: we often have to make defeasible inferences from partial information. Even if we know what utterance was made we may not know whether it was made sarcastically or not. At some stage we have to jump to a conclusion, although this may be subject to later retraction.

Secondly, although shallow systems are good at dealing with partial information, in some cases (e.g. the discussion of negation in the last section), the information they use may be more partial than it needs to be. Where we can correctly ascertain constituency (and hence scope) we should do so.

2.4 An Approach using Flat Structures

2.4.1 Choice of Flat Structure

Shallow approaches seem to gain by concentrating on information which is of interest to a specific task. To be able to do this in a deeper approach it is useful to use a representation which splits information up as much as possible. Here we will use a basic flat structure as described in Section 2 but make two extensions. The first is to allow for the same index to be used more than once, and to treat such cases as alternative readings (i.e. meta-level disjunction). For example, we take $i4:P\ i4:Q$ to mean $i4$ has the two readings, P and Q ³. We will also allow the same index to appear in more than one argument position (for

³Note that in Minimal Recursion Semantic (Copestake et al. 1995) there is a similar ability to use the same index more than once, but the interpretation is rather different. In MRS, $i4:P, i4:Q$ is equivalent to conjoining P and Q .

example, the following would be a valid set of constraints: $i4:(i1,i2), i4(i1,i3)$). These two extensions give us the ability to pack ambiguous readings in a similar manner to a chart, and to structure share similar to a chart or packed parse forest.

2.4.2 Building a Flat Structure

The extensions made above give us something akin to a ‘semantic’ chart, and allow flat semantic representation to be created directly during parsing for both fragments and full constituents. The choice to give indices names such as $i1, i2$ etc. was arbitrary, so we can equally well choose to name them e.g. $0-1-np$ (corresponding to a np edge between positions 0 and 1) or $1-2-vp$ (corresponding to a vp edge between positions 1 and 2). We merely have to ensure that if indices are used more than once, then their interpretation corresponds to the choices made above. This will be the case if we choose chart edges with the appropriate syntactic information and have a close syntax-semantics mapping. Semantics thus becomes a further annotation on the chart. Consider the syntactic chart which might be created for the sentence “US205 leave Boston”:

```
0-1-np: US205
1-2-vtr: leave
2-3-np: Boston
1-3-vp: [1-2-vtr,2-3-np]
0-3-s: [0-1-np,1-3-vp]
```

By further annotating the edges we can produce a corresponding ‘semantic’ chart i.e.

```
0-1-np: us205
1-2-vtr: leave
2-3-np: boston
0-3-s: 1-2-vtr(0-1-np,2-3-np)
```

The edge left out is the vp edge. We’ll choose to treat the semantics of the vp in a slightly unconventional way, similar to the semantics for the sentence, but with a non existing np index i.e.

```
1-3-vp: 1-2-vtr(1-1-np,2-3-np)
```

Similarly, assuming a bottom-up, left-corner or incremental parsing strategy, the fragment “leaves Boston” would get the semantic chart:

0-1-vtr: leave
1-2-np: boston
0-2-vp: 0-1-vtr(0-0-np,1-2-np)

We can think of the vp semantics as a combination of the vp with an unknown empty np. This seems to work well: we cannot infer anything about the subject of the verb but we retrain the correct relationship between the verb and its object (allowing, for example, a rule to infer that “boston” is the departure city).

Note that although the positions used here i.e. 0,1,2,3 correspond to word counts, we can just as easily use positions corresponding to time intervals, building the semantic chart directly on top of a word lattice, as is done in Verbmobil (Worm, 1999), where chart edges are similarly annotated with semantic material.

2.4.3 Applying rules to a semantic chart

What kind of rules might we apply?

Despite the equivalence between basic flat structures and corresponding logical forms, the different representations suggest rather different translation rules. Reconsider the example we had before of a pattern matching rule to interpret “at 3pm” in the context of leaving:

[leave] [at 3pm]/time(3pm) \Rightarrow departure_time = 3pm

This rule assumes some prior application of rules which determine sortal information i.e. that “at 3pm” is a time expression.

Now consider the chart/lattice we might get for the utterance “I leave at 3pm”:

0-1-np: I
1-2-vintr: leave
0-2-s: 1-2-vintr(0-1-np)
2-3-p: at
3-4-np: 3pm
2-4-pp: 2-3-p(2-2-s,3-4-np)
0-4-s: 2-3-p(0-2-s,3-4-np)

Here we have three larger edges, the first corresponding to “I leave”, the second being the pp corresponding to “at 3pm”, the third corresponding to the whole sentence “I leave at 3pm”. The semantics for “at” takes the sentence as the first argument, the time as the second. The pp arc, similar to the vp arc in the previous example includes a null first argument.

Before advocating a particular approach, let us consider more generally the kind of rules which can be applied to a semantic chart. For example, a specific rule, requiring a connected semantics for the verb phrase might be as follows (the capitalised letters stand for variables):

If we find a preposition “at” and this immediately dominates the verb “leave” then departure-time is the second argument of “at” i.e.

$$I : J(K,L) \ \& \ J:at \ \& \ K: M(N) \ \& \ M:leave \ \& \ L:T$$

$$\Rightarrow$$

$$departure-time = T$$

This rule is probably too specific, so we may want to loosen it to allow for e.g. other modifiers between the verb and the “at” phrase i.e.

If we find a preposition “at” and this dominates the verb “leave” then departure-time is the second argument of “at” i.e.

$$I : J(K,L) \ \& \ J:at \ \& \ K > H \ \& \ H: M(N) \ \& \ M:leave \ \& \ L:T$$

$$\Rightarrow$$

$$departure-time = T$$

Weakening this again we might get rid of the dominance restriction. We then have:

If we find a preposition “at” and a verb “leave” then departure-time is the second argument of “at” i.e.

$$I : J(K,L) \ \& \ J:at \ \& \ M:leave \ \& \ L:T$$

$$\Rightarrow$$

$$departure-time = T$$

This final rule is actually weaker than the pattern matching rule allowing “leave” to appear before or after “at”, and there is also no check that the argument to “at” is of sort “time”. We may want to strengthen the rule with at least a sortal check i.e.

I : J(K,L) & J:at & M:leave & L:T & time(T)
 \Rightarrow
 departure-time = T

We now have suggested several rules, the first being closest to a full semantics approach, the bottom closest to a pattern matching approach. Is one of these the correct rule?

The top rule is the most restrictive, requiring a fully connected verb phrase. The final rule is the least restrictive and thereby the most robust when encountering fragmentary input. However it is also the most likely to go wrong. For example, it would provide “departure-time = 3pm” from the sentence

I would like to leave Cambridge and arrive at 3pm

The rule thus only makes sense in the context of other rules which can override it., e.g. a more specific “arrival-time” rule which worked on the verb phrase “arrive at 3pm”. In pattern matching systems this would typically done via temporal ordering: more specific rules are applied first and block weaker rules from applying.

This discussion suggests that no one rule should be chosen. Instead we need a range of rules for “departure-time” and “arrival-time” some of which are more specific than others. In a particular scenario we should pick the most specific rule which can apply.

To compact up the rules, the approach taken here is to provide a set of obligatory constraints, and some optional constraints. The four rules above are compacted into the single rule:

Obligatory constraints: {I : J(K,L), J:at, M:leave, L:T, time(T)}
 Optional constraints: {K > H, H: M(N), K : M(N)}
 \Rightarrow
 departure-time = T

A particular application of a rule will get a weight according to the number of obligatory and optional constraints which hold.

Inference, Underspecification and Translation

The rules above are expressed as inference rules i.e. “if we have A and B then we can infer C”. This suggests that getting the value for a slot is a process of inference from

a semantic representation which may be underspecified (the rules do not check that the representation is fully specified). The inference is also defeasible, since there is no checking of the monotonicity properties of the surrounding context, and various assumptions are missing. Another way to think about this is in terms of Abduction (c.f. Hobbs et al. 1993) An inference could go through if we added in various extra constraints, and we want to add in as few extra constraints as possible (hence more specific rules are preferred). Note that this is a non-standard use of Abductive Reasoning since we are allowing abduction of information not just about the context etc. but also about the actual structure or content of the utterance.

A defeasible inference perspective works well when we consider single slot values. However, when we consider multiple values, or items in the scope of other operators, it is useful to think in terms of translation. The idea of treating database enquiry as translation goes back to at least Bronneberg et al. (1980), and the use of inference for translation into database slots was advocated by Rayner (1993). Taking a translation approach, we can consider translating individual pieces of the utterance, and then put the pieces back together again. This should allow better generalisation. It also gives us a better story as to why a more specific rule for e.g. arrival time would block a less specific rule for departure time (even if higher weighted, or applied earlier). Once a term has been translated, it will not be available for translation using another rule. To rephrase the rules above as translation rules we have to distinguish between items to be translated and the surrounding context. For example, the departure-time rule would become:

Constraints for translated terms: {time(T), J:at, L:T, I:J(K,L)}
 Constraints for rest of utterance: {M:leave}
Optional constraints: { K > H, H: M(N), K : M(N)}
 New constraints: {P: translation(J,L,Q), Q:departure_time=T}

Here the two indexed items, J and L (corresponding to the word “at” and the time) are translated to the term Q which is a statement that the departure time has the value T. Other items such as the M (corresponding to the word “leave”) are constraints on the rule, but are not part of the translation.

Now consider an example containing a negation e.g.

Not to London on Thursday, to Manchester on Friday

The negation can be translated separately e.g.

Constraints for translated terms: {I:J(K,L), J:not}
 Constraints for rest of utterance: {}

Optional constraints: {}

New constraint : {R: in-order(-M,+N), P:translation(K,M), Q:translation(L,N)}

This states that there is a translation of “not” as a database command which first checks that the constraint corresponding to M does not hold, then asserts the constraint corresponding to N (note that this is just one of the many possible translations of “not” ; alternative translations are required for “not via London”, or “not Boston, London”).

This rule is appropriate if the material dominated by the “not” has a connected parse, with the immediately dominated material having a translation. We can weaken this to allow dominance rather than immediate dominance or backoff further e.g. to allow the first argument of “not” to be anything upto the next comma⁴.

How does this translation approach differ from the defeasible inference approach? In this approach we translate all the terms we think are important, and assume that all other terms are semantically empty (null constraints or identity functions). It is this assumption that is defeasible, along with relaxations allowed within the rules.

For completeness, it should be mentioned that it would be possible to allow rules which access the values of translated subconstituents i.e. a rule may fire if subconstituents have appropriate translations. This however does introduce rule order dependence: we would have something very similar to cascaded pattern matching, where first items are annotated as e.g. companies and losses, then a rule picks up the companies and losses to annotate the sentence as a whole as a company loss.

Finally we should note that things get more complex when we consider alternative readings for the same section of utterance (which of course is the normal state of affairs when we start with a speech lattice). In this case we cannot be sure of the surrounding context, and we do not want translations of different parts of the utterance to depend upon different readings of the context. This means that assumptions need to be preserved (e.g. the contextual constraints above), and if there are cascaded rules, the assumed constraints need to filter into the result (in logical terms this is just a case of $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$). We will consider one possible architecture at the end of this paper, but this is by no means the only possibility.

⁴This kind of constraint is actually a syntactic constraint, not a constraint about the content or structure of the semantic representation. However, there seems no problem with mixing syntactic and semantic constraints in this kind of approach where everything can be seen as information attached to the same chart.

Translation Rules for a Semantic Chart

The final change we need to make to the rules is to allow for the effect of the context of the previous utterance, and the dialogue contribution so far (e.g. what slots have already been filled etc.). The translation rules are thus partly dealing with semantic information, partly with what might traditionally be treated as pragmatic information. An example translation rule is the following:

Constraints for translated terms: {time(T), J:at, L:T, I:J(K,L)}
Constraints for rest of utterance: {}
Constraints from prior utterance: {M:query(departure_time)}
Contextual slot values: _
Optional constraints: {}

New constraints: {P: translation(J,L,Q), Q:departure_time=T}

This is appropriate for picking out the departure time where the question asked what the departure time was, and the answer contains a time. This rule will fire provided there is no more specific rule to translate the terms to something else.

Now let us reconsider the various requirements we had earlier and see how this approach fits.

2.5 Evaluation of the Flat Structure Approach

Robustness

The approach outlined above has most similarity in dealing with fragmentary input to pattern matching approaches. There is no need to try to reconstruct a full semantic analysis, instead a weaker version of the translation rule is used (with fewer optional constraints applying). Where there is ambiguity, more specific rules are preferred.

Domain Dependence

Domain dependence is built into the translations rules, rather than there being any particular filtering stage based on domain specific preferences. Different readings are preserved until the translation rules fire. Although a sentence such as “Show flights to Boston” will

get two readings, there is no need to choose between them. The translation rules (similar to pattern matching rules) will only pick up on the appropriate possibility. More empirical research is required to see if this will work in all cases, or whether there might sometimes be a role for domain specific preferences to adjust the weightings given to different components.

The preservation of different readings until translation allows correct treatment of the “Boston London Heathrow” case. Finally, we should note that there is no a priori preference for longest fragments or the shortest path through the chart. Preference is given to more specific instantiations of rules. Larger fragments may well result in more optional constraints being satisfied. However this will only be if the larger fragment introduces more material which is relevant to the task: the addition of a bogus word will not be rewarded. Moreover, specificity gained in this way still has to be balanced against specificity gained from contextual constraints.

Context Dependence: Previous Utterance

Contextual dependence is included by adding constraints to the translation rules. This can comprise constraints on the previous utterance (e.g. which slots are being queried), or on the current dialogue state. In simple task oriented dialogues the state is likely to be just a set of slots, some of which are filled. Contextual constraints count towards making the rule more specific, though to a lesser extent than information within the utterance itself. The aim is for the top down expectation provided by contextual constraints to be able to affect the interpretation without overriding what the utterance actually says.

Context Dependence: Other parts of the utterance

The approach here is similar to shallow processing in allowing the translation of one part of an utterance to be affected by the translation of other parts.

Reconfigurability

Reconfiguring to a new task requires the introduction of new translation rules, and the addition of lexical entries for at least the words mentioned in the translation rules. The robust nature of the approach means that we can provide a working system without providing a full parse for all, or even for a majority of the utterances in the domain.

Accuracy

The aim of this approach was to give the robustness you might expect from shallow approaches, but use as much linguistic information as available to ensure accuracy is as good as possible. The use of rules which can be more or less specific should achieve this. Where parsing provides appropriate constituency (and hence appropriate scope for operators such as “not”) this information is used. Where not, the relaxed versions of the rules should at least equal the results of shallow processing.

2.6 A preliminary implementation

Most of the ideas in this paper have been integrated into a prototype dialogue system designed for route planning. The system allows for simple question answer dialogues such as the following:

S: Where would you like to go?
U: London
S: Where are you leaving from?
U: Cambridge
S: Would you like the quickest or the shortest route?
U: shortest
S: When would you like to arrive?
U: Before five p m.

However, it also allows for a user to supply more information than expected or different information. For example, the following exchange is more appropriate for an experienced user of the system.:

S: Where do you want to go?
U: I would like the shortest route from Cambridge to London to arrive before
5

The system builds a chart using a fully incremental parser based which adds new edges or annotations to extend a word lattice, Categorical Grammar was used to give a straightforward syntax/semantics interface, and the grammar was subsequently compiled into simple Dependency Grammar. This enabled the use of a packed parser based on the packed incremental recogniser described by Milward (1994).

The present algorithm applies each translation rule to the chart, and picks up a set of potential slot-value pairs (for the slots, destination, source, mode, and arrival/departure time). Each slot-value pair is weighted according to the specificity of the rule used. The weighting is pretty crude, and achieved by adding the number of items mentioned in a rule, with higher weightings for closer items. More advanced weighting schemes based on Information Retrieval technology (where more interesting terms get higher weights) can be imagined. When spoken input is used we would also want to use the recogniser weightings.

After creating the set of potential slot-values, the algorithm then filters the set to obtain a consistent set of slot-value pairs. The first stage is to filter out any cases where the translated material overlaps. In these cases the more specific translation is retained, the others are dropped. Secondly there is a check to ensure that no slot is given the same value twice. If there is a conflict the higher weighted value is adopted.

It should be noted that the current algorithm does not check consistency of contextual constraints: different assumptions about the context may have been made during the filling of different slot values, and ideally the algorithm should check that each translation corresponds to a consistent path through the lattice. Despite this, the ability of the rules to deal with context and domain dependency, and to provide robust interpretation seem to give the system creditable performance.

2.7 Conclusions

This work points to various ways in which we can mix some of the advantages of linguistic analysis with shallower methods. The approach advocated incorporates linguistic information where necessary (e.g. for determining the scope of negation), but also allows linguistic constraints to be relaxed to ensure robustness.

Chapter 3

Robust Semantic Processing of Spoken Language: Combining Fragments

3.1 Introduction

In this chapter we present the approach to the robust analysis of spoken language dialogues that is implemented in the Verbmobil system. The processing of spoken language and, hence, the results of speech recognition, provides the main criteria for determining what robust processing ought to be able to handle. As will be seen this leads us to an approach that is specifically oriented to observed phenomena in spoken dialogues. The fact that Verbmobil is a translation system provides important secondary criteria for the behavior that the robustness strategy must maintain, since we are processing dialogues between humans in which the system is a mediator and not a participant.

We commence our presentation with some remarks on robustness in general and as applied to the processing of spoken language. We then describe the three key components of our approach: the local system architecture, the notation for the description of robustness rules, and the processing model that applies these rules. We proceed to give some examples of the kind of rules that are defined for the Verbmobil system. We conclude with some evaluation results for the most significant criterion in a translation system: the contribution to translation quality, and with a summary of the salient properties of the approach.

3.2 Robustness

3.2.1 Robustness and Graceful Degradation

A natural language processing system is called *robust* if it does not completely fail when confronted with ungrammatical or unexpected input, but rather its performance *degrades gracefully* (Stede, 1992, Menzel, 1995). The ungrammaticality in the input may be caused by the speaker, as the input may contain spoken language phenomena or other ungrammaticalities, or by the system itself, as the speech recognizer may deliver output containing recognition errors.

A computer system exhibits a graceful degradation behavior if, when encountering a situation where an unexpected error occurs, it does not collapse, but its level of performance may be diminished, depending on the severity of the error (Beardon, 1989).

Graceful degradation means in the context of spoken language analysis that if the system cannot process its input completely, it will nonetheless provide some kind of result. This can mean that the analysis is less fine-grained and detailed, or as an alternative, a more liberal processing strategy may have been used.

In summary, a robust system delivers a result even for an input not matching its expectations. The quality of this result will depend on the degree to which the input deviates from the original expectations. If the deviation is only slight, the result will be (almost) as good as if no deviation occurred. The more the input deviates from the standards, the lower the quality of the result will be.

3.2.2 Robust Processing of Spoken Language

It is this relative notion of robustness that we have attempted to implement in a system for the translation of spoken dialogues. The functionality that is to be maintained is the linguistic analysis of the results of speech recognition. In this context there are two main perturbing factors: spontaneous speech phenomena leading to ungrammaticalities in the spoken input and errors in speech recognition. While these are quite distinct influences, the effects may not in all cases be distinguishable. We could make an analogy and think of the input as a signal, a *linguistic* signal, with a specific package of information to convey. The perturbing factors are then noise on that signal and the task of robust processing is to convey as much of the information as possible.

It should be noted that in the Verbmobil context, where we deal with the translation of

the utterances of two (or more) human speakers, it is not practical to prioritize the pieces of information a linguistic signal may carry. In contrast, dialogue systems concerned with human computer interactions have, indeed need, much stronger dialogue models which provide expectations as to the information the system requires next. This allows robust behaviors to be defined as seeking specific types or pieces of information (c.f. Milward,1999) although this may imply more than one simultaneous search.

In a human-human dialogue where the system has far less control over what is spoken it is easier and, we assume, more efficient to define robust behaviors in terms of specific linguistic phenomena, based on observed examples. This is an incremental, prototyping process and is, by definition, both heuristic and incomplete. However, it does permit the maximum vestigial functionality for those phenomena that are covered. The implication for the graceful degradation curve is that it starts very close to idealized behavior but plunges after a while into an abyss because either no phenomena or too many phenomena are recognizable. These are properties that are inherent in the approach we adopt, but this represents a choice based pragmatically on observed behavior in the current *Verbmobil* system and its predecessors, and on expectations about the frequency of the phenomena we intend to be robust against.

3.2.3 Robust Semantics Processing

The operations that are intended to provide a degree of robustness in the linguistic analysis are factored out of the main parsing and semantic construction components. The reasons for this separation are explored in more detail in the next section, but a consequence of this is that the rules for robustness apply over *Verbmobil Interface Terms* (VITs) which, although they combine various types of information, are primarily semantic representations. There are certain advantages to addressing robustness at a more abstract level of representation, such as the fact that abstracting away from details of configurational syntax or morphosyntax becomes trivial, since this information is only represented if required. Most of the operations that are defined in the robustness rules are essentially semantic in nature, affecting the content of the representation, though the tests on which they depend may combine various types of linguistic information. So robust semantic processing is semantic in the sense that it operates over a level of semantic representation.

However, this type of semantic processing does not draw on any deeper inferential or contextual semantics. Just as the operations applied for robustness are not tailored to obtaining specific information to meet the expectations of a dialogue model, so the operations do not attempt to test which of the possible repaired propositions is the most appropriate for the predicted current dialogue act or the most consistent with the local context. The rules are instead formulated in purely linguistic, i.e. syntactic and semantic terms. For the phenomena that have been addressed this was, in general, adequate.

In short, the robustness rules are driven by formal linguistic properties of the phenomena they are intended to deal with as apparent in the, possibly partial, analyses available, so they cannot be expected to, in any sense, understand the content of either the input they receive or the results they make out of it. This level of semantics, like the management of the dialogue, is left to the human users.

3.3 Architecture

The architecture in which robust semantic processing was implemented was determined by two sets of constraints: the nature of the search problem in word lattice parsing and the inherent time constraints necessitated by an application in a spoken language dialogue system¹. The decision to implement robust behaviors as postprocessing in a subsequent module stems from the primary concern of providing the best possible analysis of the input. Once this partition was made the extension of the architecture with additional parsers was primarily a technical decision.

3.3.1 Accurate and Robust Parsing of Word Lattices

The global goal of the linguistic analysis components in Verbmobil is to present the Transfer component with the most accurate representation of the spoken input they can produce in the available time. The starting point is the annotated word lattice, or *Word Hypothesis Graph* (WHG), provided by the speech processing components. The parsing of a word lattice differs in a number of respects from parsing in a text-based system, but one of the most crucial is that there are essentially two decisions to be made, rather than just one. The parser must determine which sequence of words provides the best analysis as well as which analysis to assign. This implies, in effect, that there must be choice points where one path through the lattice is rejected and processing continues on the next most likely path.

Rather than complicate these decisions with a third possibility of processing the original path by alternative means, we factor out the robustness rules into a postprocessor which systematically applies to the intermediate results of parsing. The most viable sequence of partial analyses is presented on a path by path basis, as reported in Kaspar et al.(1999). In effect, we give priority to the accurate analysis of the input lattice with a restrictive grammar and provide the results of robust processing as a fallback position should no full

¹In previous presentations we have emphasized the time constraints but these are conditions that we would, in any case, have to come to terms with.

analysis be found. In this way we attempt to ensure that, if at all possible, a full analysis of a grammatical string will be found.

There are other benefits for the linguistic adequacy of robust semantic processing that derive from its role as a postprocessor on linguistic analyses, such as the fact that various different types of linguistic information are available. This is an advantage to the formulation of robustness rules as heuristics that may be triggered by various types of information not necessarily accessible in a local compositional analysis. The fact that our methodology for the definition of robustness rules is data driven and example-based is explicitly coordinated to the fact that we have full access to analysis results and the information they contain.

3.3.2 Efficiency and Time Constraints

The architecture we have developed for coordinating the robust processing of spoken language with more standard analysis techniques places a high priority on the ability to achieve a result within the rather strict time constraints imposed by the task domain of processing spoken dialogues. We assume that the user will generally expect a response within some small multiple of the speech time for the input.

There are essentially two ways in which the local architecture can help to meet such stringent performance constraints. On the one hand, by being prepared to produce a result on demand, irrespective of the internal status of the processor. This readiness to provide a result whenever it is required is often termed the *anytime property*, but it does not presume that at any time the quality of the result will be equal, quite obviously more processing time may lead to better results.

The second and more substantial contribution is an efficient distribution of labour between the core functionality of linguistic analysis and the additional robustness behaviors provided in our module. The definition of a separate module is part of this division of labour, in a literal sense, but also because it makes it easy for a restrictive analysis to proceed, largely, in parallel with robust processing. This is also facilitated by the fact that the analysis modules are able to provide partial, intermediate results for fragmentary analyses as well as full spanning analyses when these are available.

As a direct consequence of this division of labour and the separation of parsing and robust analysis into separate modules, it becomes feasible to employ multiple parsers, also operating in parallel.

3.3.3 Multiparser Architecture

A distinctive feature of linguistic analysis in the Verbmobil system is that several analysis techniques are employed. The assumption behind this is that the various analysis techniques will have different strengths and weaknesses, so that the result of combining them will consolidate the coverage of the domain. This should ensure that a high proportion of grammatical constructions are covered by at least one parser and that an analysis can be assigned quickly by the most appropriate parser.

Figure 3.1 shows a simplified representation of the Verbmobil architecture in which only the components involved in translation based on linguistic analysis are shown. The modules concerned with linguistic analysis occupy the central area of the diagram, from the input of annotated WHGs into the integrated processing module to the output of the resolved VITs from the semantics module, of which robust semantic processing is the main submodule.

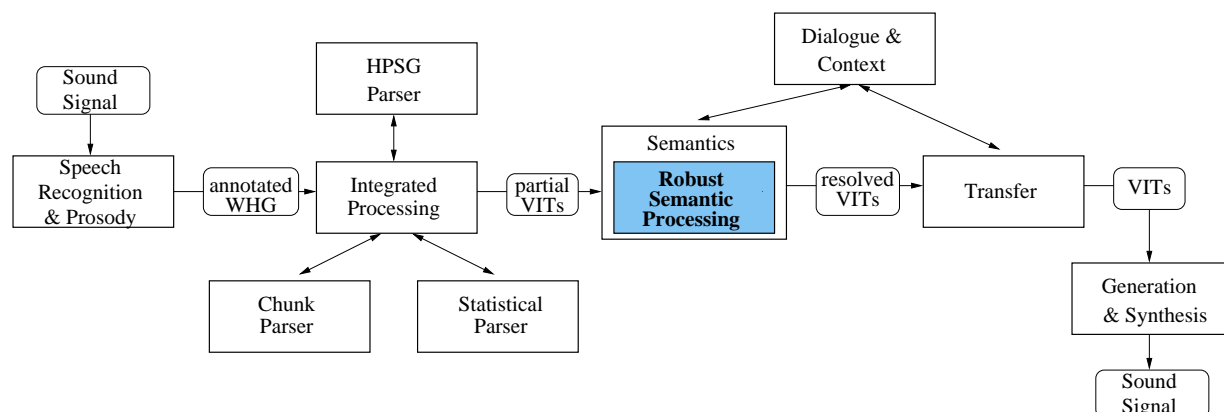


Figure 3.1: Part of the Verbmobil system architecture.

The partitioning of the basic architecture for linguistic analysis into a restrictive parser and a postprocessor for robustness rules facilitates the use of more than one parser. Provided that the interfaces are maintained and the full and partial analyses derived by additional parsers are formally the same type of object, then the robust processing module does not actually need to know where its inputs come from. While, in this sense, all parsers are treated equally, as they are depicted equally in the architecture diagram, there are differences in both the quality of the results they produce and their general input and output behavior. The differences between the parsers are, indeed, the very motivation for employing more than one.

3.3.4 The Various Parsers

The parsers that are employed are:

1. A chart parser (see Kiefer et al., to appear) using an **HPSG** grammar and the corresponding typed unification formalism. This provides the most detailed, theoretically motivated analyses.
2. A **statistical** LR-parser that uses a context free grammar (see Ruland, to appear) derived from a tree bank of (hand) analyzed utterances from the Verbmobil corpus.
3. A **chunk** parser based on cascaded finite state automata (see Hinrichs et al, to appear), but augmented with a direct lookup in the tree bank.

While the HPSG parser quite naturally provides a full semantic representation in VIT format, the other two parsers are essentially tree parsers and require a pass through a semantic construction module (see Schiehlen, 2000) to construct the corresponding VITs. Some morphosyntactic information, such as case and agreement feature, was excluded from the tagset on which the tree bank analyses are based, so that this information is also missing in the statistical and chunk parser analyses.

The chunk parser is the most likely to produce a result, though this may be a sequence of fragments. This is also the quickest of the three parsers. The HPSG parser is the only parser that will return more than one analysis for a given input, where these may be distinct analyses for the same sequence of words, or initial fragmentary analyses for ungrammatical paths followed by a spanning analysis on a subsequent path.

3.4 The Description of Robustness Operations

Our primary strategy for providing robustness against the effects of spontaneous speech phenomena and recognition errors depends on the definition of rules which operate on the sequence of, possibly partial, analyses received from the parsers to produce a more adequate analysis. Each rule is intended to detect a specific phenomenon, by whatever means is necessary, and define an action appropriate to that phenomenon. The classification of phenomena is based on observed behavior of lattice parsers during the development of the system. The rules are therefore, in essence, example-based. They are also heuristic, since it is only assumed that the action appropriate in the example will be effective in another context. For this reason, confidence values are associated with the rules, or rather with the

operations they incur. This allows the relative ranking of robustness rules and the ranking of results from robust semantics relative to those from a parser.

Here we discuss the levels of information which are relevant for the task of robust semantic processing and the general rule syntax. A survey of the rules actually used is given in section 3.6.

3.4.1 The Relevant Levels of Information

Dealing robustly with input which contains spontaneous speech phenomena or is ungrammatical due to recognition errors or performance phenomena requires information from all level of linguistic description:

Prosody: Prosody provides important clues to the correct interpretation of spoken language. For robust analysis, the most important information from prosody is information on sentence and phrase boundaries within an utterance, but prosody can also signal the presence of self-corrections.

Syntax: When confronted with fragmented input, the syntactic properties of the fragments provide important information on their combinability. E. g., a verb has a subcategorization frame stating the requirements it imposes on potential arguments. However, in order to deal robustly with corrupted input, it may be necessary to allow for violation of such constraints.

Semantics: The semantic properties of fragments also determine their combinatoric potential. E. g., verbs require that their arguments be of certain sorts. If a potential argument has the sort the verb requires for an unbound argument position, this position might be filled with the argument even if a constraint from the syntactic level such as case requirements is violated.

Discourse: Spoken utterances occur in the context of a discourse, and a spoken discourse usually has a certain structure from which predictions about what may be uttered next can be made. Such predictions can then be used to evaluate hypotheses about the following utterance if it cannot be fully analyzed.

If all these levels of linguistic analysis are required in dealing with ungrammatical input, a representation is required which combines information from all these levels. What is of special importance are the links between the levels. E. g., if a verbal semantic predicate is missing an argument, one needs to know the syntactic subcategorization frame and the case and sort of a potential argument.

3.4.2 Semantic Representations: The VIT Format

The VIT is the representation format used at all interface between linguistic modules in the Verbmobil system, see Schielen et al.,(2000). It combines a semantic representation with a wide range of other types of linguistic information which were found to be useful in cross-linguistic processing.

The semantic components of the VIT can be viewed as a theory-independent representation for underspecified semantic representations (Bos et al, 1996, Pinkal, 1995). It specifies a set of discourse representation structures, DRSSs, Kamp and Reyle (1993). If an utterance is structurally ambiguous, it will be represented by one VIT, which specifies the set of DRSSs corresponding to the different readings of the utterance.

All semantic predicates in a VIT are uniquely labeled. This allows the the VIT to be a completely flat structure, avoiding any recursion, as opposed to DRSSs, which can contain sub-DRSSs. In order to express substructures corresponding to sub-DRSSs in a VIT, the labels of the predicates in question are associated in a so-called *group*, which can be seen as a logical conjunction of its member predicates. E. g., groups can be in the scope of operators.

Technically, a VIT is represented as a nine-place PROLOG (Clocksin and Mellish, 1994) term. There are slots for an identifier for the input segment to which the VIT corresponds, a list of the core semantic predicates, a list of scopal constraints, syntactic, prosodic and pragmatic information as well as tense and aspect and sortal information. A VIT contains information from all levels of description which are relevant for robust analysis of speech.

For creating VITs, accessing the information in them, and checking the formal well-formedness of a VIT, an abstract data type has been implemented (Dorna, 2000). An example of a VIT for the sentence *Wir müssen ja noch einen Termin vereinbaren* ('We still have to arrange an appointment') is given in figure 3.2.²

3.4.3 The Rule Syntax

The rules describe three kinds of information:

1. The conditions under which they are to be applied,
2. the operations that should be performed, and

²Note that the sentence has a slightly different meaning if the *noch* is stressed; then it should be translated as 'We have to arrange another appointment'.

```

vit( vitID(sid(102,a,ge,0,700,1,ge,y,syntaxger), % Segment ID
      [word(wir,r0,[lh159]), % WHG String
        word('m"ussen',r1,[lh161]),
        word(ja,r2,[lh164]),
        word(noch,r3,[lh167]),
        word(einen,r4,[lh170]),
        word('Termin',r5,[lh174]),
        word(ausmachen,r6,[lh176]))],
      index(lh157,lh156,ih158), % Index
      [decl_imp_quest(lh179,hh178), % Conditions
        pron(lh159,ih160),
        muessen(lh161,ih158,hh162),
        ja(lh164,hh165),
        noch_scop(lh167,hh168),
        ein(lh170,ih171,lh172,hh173),
        termin(lh174,ih171),
        ausmachen_unspec(lh176,ih177),
        arg3(lh176,ih177,ih171),
        arg1(lh176,ih177,ih160)],
      [in_g(lh179,lh157), % Constraints
        in_g(lh176,lh175),
        in_g(lh174,lh172),
        in_g(lh170,lh169),
        in_g(lh167,lh166),
        in_g(lh164,lh163),
        in_g(lh161,lh156),
        in_g(lh159,lh157),
        leq(lh175,hh162),
        leq(lh175,hh165),
        leq(lh163,hh178),
        leq(lh175,hh168),
        leq(lh166,hh178),
        leq(lh169,hh178),
        leq(lh175,hh173),
        leq(lh156,hh178)],
      [s_sort(ih160,human), % Sorts
        s_sort(ih158,mental_sit),
        s_sort(ih171,;(time,meeting_sit)),
        s_sort(ih177,;(mental_sit,communicat_sit))],
      [prontype(ih160,sp_he,std)], % Discourse
      [pers(ih160,1), % Syntax
        num(ih160,p1),
        gend(ih160,;(masc,;(fem,neut))),
        gend(ih171,masc),
        num(ih171,sg),
        pers(ih171,3)],
      [ta_tense(ih158,pres), % Tense and Aspect
        ta_perf(ih158,nonperf),
        ta_mood(ih158,ind),
        ta_perf(ih177,nonperf)],
      [] % Prosody
    )

```

Figure 3.2: A VIT for the sentence *Wir müssen ja noch einen Termin vereinbaren.* ('We still have to arrange an appointment').

3. what the result of a rule application should be.

Since the robust semantic processing has been implemented in PROLOG (Clocksin and Mellish 1994), we define the rules in PROLOG-style syntax. The general format of a rule is given in (3.1).

```
(3.1) RuleID ::  
      ListOfConditions ---> ListOfOperations & OutputVIT.
```

For ease of reference, each rule comes with a rule identifier `RuleID`.

`ListOfConditions` is a list of conditions on the input VITs to which the rule may be applied. The order of conditions matches the temporal order of the input VITs. While there may be empty gaps between the input VITs, there should be no VIT constituent in between.³ To express a conjunction of several conditions which should be fulfilled by a single input VIT, these conditions are grouped as a list. To express dependencies between several input VITs PROLOG variables may be used. To express the negation of a condition or a sequence of conditions (stated as a list), the operator `not` may be used.

`ListOfOperations` is a list of operations which should be applied to the input VITs. The order of the list elements defines the order in which the operations should be applied. The application of the operations takes place only when all input VITs have been identified, i. e., when all variables for VITs have been bound.

`OutputVIT` simply defines which VIT should be the output of the rule. The output VIT may be one of the input VITs or the result of an operation in `ListOfOperations`.

```
(3.2) RuleExample ::  
      [[cond1(V1),cond2(V1)],prop(V2,X),prop(V3,X)] --->  
      [op(V1,V3,V4)] & V4.
```

Consider the example rule given in (3.2). It requires three input VITs, `V1`, `V2`, and `V3`. `V1` should fulfill the conditions `cond1(V1)` and `cond2(V1)`, while `V2` and `V3` both should have the same value `X` for property `prop`. If this is the case, the operation `op` is applied to `V1` and `V3`, delivering `V4` as a result, which is also the result of the rule.

³If the bridging mechanism (see Section 3.5.2) is applied, VITs may be left out when applying rules.

3.4.4 Confidence Values for Operations

The robustness rules are triggered by phenomena that occur with varying frequency and the solutions that they offer may vary in effectiveness. It is therefore necessary to associate a confidence value with each rule so that the selection routine (see Section 3.5.3) can correctly evaluate each contribution made by robust semantic processing to the overall analysis. In practice, the confidence values are associated with the operations carried out when the rule is successfully applied. The confidence value for the rule is the product of the values of each of the operations it applies.

The confidence value of an operation will be influenced by the degree to which linguistic constraints are obeyed or violated. If a verb is combined with a potential argument, the confidence value has to reflect how many of the verb's requirements are met by the argument, e. g. case, sort, person, number, etc. Similarly, operations of a more heuristic nature need to be assigned a low confidence value in order to favor the results of more reliable operations, if those can be obtained.

3.5 The Processing Model

The processing model we propose divides into three phases of processing, of which the first and the second are performed in alternation while the parser analyses the input and delivers intermediate results, and the third is performed at the end of processing:

1. *storing* the (possibly partial) analyses for different WHG (sub-)paths delivered by the parser;
2. *modifying and combining* partial analyses to yield bigger structures by applying a set of rules; in some cases, *deleting* partial analyses;
3. *choosing* a sequence of (possibly still partial) analyses from the set of hypotheses as output.

First, the (possibly partial) results delivered by the parser for different paths or subpaths in the WHGs are stored. Second, partial results are combined by applying a set of rules to them. This may involve the modification of analyses in some cases, e. g. if type coercions are performed, or even deleting some partial analysis if a spoken language phenomenon such as a self-correction is hypothesized. The analyses resulting from modification or combination are also stored and are then subject to further combination. These first two

phases of processing take place while the parser is still searching for an analyzable path in the WHG; each time a partial result is built up by the parser, it is processed in this way.

Third, when no spanning result could be found by the parser, or the time limit for the analysis is reached, a result has to be selected from the partial analyses. In the best case, there is a spanning result covering the whole input segment; otherwise, it will be a sequence of partial results, each covering a part of the input segment.

3.5.1 Storing Partial Analyses

We chose to model the process of storing and combining partial parser results in robust semantic processing following the model of chart parsing (Kay,1980). This is quite natural, since the combination is done by applying rules to the partial results, which each cover a certain portion of the input, like the edges in a parser's chart. However, in this task there is no predefined goal state for processing. In the general case, it is not possible to reach a completely saturated analysis; many utterances are, by their nature, incomplete. Thus, the decision as to what counts as a result, cannot be encoded in the rule set but has to be taken in a separate selection mechanism. It follows that the resulting chart is a directed graph whose edges are labeled with VITs which may or may not form part of the final analysis. On analogy with the WHG we terms this a *VIT hypothesis graph* (VHG).

Robust semantic processing therefore proceeds as a forward search, applying its rule set to the input analyses and the intermediate results derived from them, as long as rule applications are possible and there is processing time available. The decision when to retrieve the results is external to robust semantic processing.

The number of analyses is potentially large, thus redundancy should be avoided wherever possible. In an application with multiple parsers, even multiple equivalent results for the same parts of the input have to be handled. Chart parsing algorithms avoid redundancy both in storage and processing.

The chart parsing method lends itself very well to the incremental mode of processing we employ in robust semantic processing. New edges can be added to a chart basically at any time. Furthermore, the use of an agenda makes it possible to assign different priorities to results from a parser and from robust semantic processing and to interrupt processing to give preference to new input.

For application in a speech processing system, constraints on the processing time available play an important role. The chart and agenda-based mechanism is suitable for time-constrained processing, since results can be retrieved from the chart as soon as the first edge has been entered into it. The more processing time is available, the more results will

be produced and entered, and the more combinations will have been performed, increasing the quality of the result.

3.5.2 Combining Partial Analyses

The combination of partial results is done on the basis of a set of combination rules. Their application to edges entered into the chart results in new edges. The rule application and the adding of edges to the chart is controlled by an agenda mechanism. In many cases, there may be gaps in the input or edges which cannot be integrated into a complete analysis. This kind of situation is handled by the bridging mechanism which selectively excludes some of the edges stored in the chart.

Rule Application.

The application of combination rules in our model follows that in the chart parsing scheme. Each time a passive edge is entered into the chart, an edge is added to the agenda for each rule whose first argument matches the new analysis. For unary rules these are passive edges, otherwise they are active edges.

Each time an active edge is taken from the agenda and added to the chart, all possible combinations with passive edges already present in the chart are computed and the resulting active and passive edges are added to the agenda.

The Agenda.

We carry over the concept of an agenda from chart parsing to robust semantic processing. The agenda is a data structure that stores new edges to be added to the chart. Edges are taken from the agenda, added to the chart, and the new active and passive edges induced are again added to the agenda.

The agenda administers the introduction of both active and passive edges, since new edges arise not only from local processing but also as new results from the parsers. The selection of the next edge from the agenda to be entered into the chart implements an execution strategy which exhibits the following priorities:

1. Passive edges from a parser are processed first.

2. Passive edges are processed before active ones.
3. Passive edges with higher scores are processed before other passive edges.
4. Longer active edges are processed before shorter ones.

The agenda mechanism furthermore supports the distributed processing idea behind robust semantic processing, which operates in the background while the parser is searching for an analysis. Intermediate results are handed over to robust semantic processing and added to the agenda. The agenda is processed item by item. Before an edge is taken from the agenda, a check is performed whether a new analysis from the parser has arrived, which would then be handled with precedence.

The Bridging Mechanism.

Robust semantic processing has to be more liberal than parsing in order to achieve an integrated analysis even if the parser cannot find a spanning result. One sense in which it has to be liberal is that it has to allow for gaps when constructing an integrated result.

Gaps may occur in two ways. First, there may be no partial result for a certain interval of the input. This may be due to a pause the speaker made, or there might be a sequence of words recognized which the parser cannot analyze in any way.

Second, there might be words or phrases which are analyzed but cannot be integrated into a spanning result. This may be due to a self-correction, i. e. the speaker uttered a word or words not meant to be part of the utterance, or due to speech recognition errors, resulting in fragmentary analyses. In the case of a self-correction, there may be a prosodic marking indicating that a self-correction occurred. This can help in determining which interval should be left out of the analysis. The bridging mechanism is supposed to deal with both kinds of situations. An interval should be bridged if it is empty or contains material that cannot be integrated.

Bridging means the extension of active edges in the chart from left to right. More precisely, if there is an interval to bridge, copies of all active edges ending in the node where this interval to bridge starts are added to the agenda. The copies contain the same information as the original active edges, except that they end in the node where the interval to bridge ends.

Consider the VHG shown in Figure 3.3⁴. The sequence of words, *ich treffe habe Termine* ('I meet have appointments'), is not well-formed, but it exhibits a pattern typical of false

⁴Active edges are shown as dotted edges here

starts. The analysis *ich habe Termine* ('I have appointments'), requires the bridging of the redundant verb *treffe* by extending the active edge induced by the pronoun *ich*.

Since it takes a certain amount of time before a critical mass of analyses have been accumulated from the parser, the chart will contain gaps most of the time. Therefore, the bridging mechanism only applies when the parser has stopped working because the time limit has been reached, since at that point, it is clear that no further analyses will be produced which could fill the gaps. At this moment, it is also clear which analyses cannot be integrated into a result and thus need to be bridged.

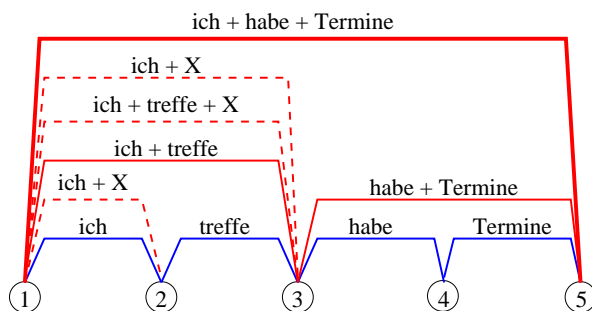


Figure 3.3: An example for the bridging mechanism

3.5.3 Scoring and Result Selection

At the end of processing, the chart will usually contain a significant number of competing hypotheses. A result has to be selected from these hypotheses. There can be two kinds of situation:

1. There is at least one analysis which covers the complete input; if there are more than one, one of them has to be chosen.
2. No spanning analysis is present; a sequence of partial results has to be selected.

What is needed in both cases is a scoring functions which aids the search process for a result by assigning a score to each hypothesis.

The Scoring Function

The scoring function σ plays a central role in the selection of results. It has to assign each edge, whether delivered by the parser or built up by robust semantic processing, a score

which reflects its relative quality and reliability as closely as possible.

The scoring function has to take into account several bits of information about an edge. Furthermore, it has to fulfill certain requirements which follow from intuitions about the functioning of robust semantic processing:

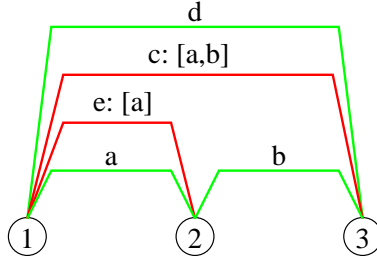


Figure 3.4: Scoring edges from different sources

1. Parser results are better than robust ones.
2. Integrated results are better than fragmented ones.
3. Longer analyses are better than short ones.

Figure 3.4 is adapted from Worm (2000) to illustrate how these preferences are incorporated into the scoring function defined below. The two basic cases distinguish parser edges, such as d , which have no component edges from the robust edges, such as c , which has components a and b .

Definition 1 (Scoring function σ) For an edge e , with component edges $C(e)$, length⁵ $|e|$, where $PC(e)$ and $RC(e)$ are the confidence values for parser analysis and robust rule respectively, $RN(e)$ is the (normalized) reading number, $GP(e)$ the number of glue predicates, and $UA(e)$ the number of unbound arguments positions.

$$\sigma(e) \stackrel{def}{=} \begin{cases} (|e|^2 \cdot PC(e)) - RN(e) - GP(e) - UA(e) & \text{if } C(e) = \emptyset \\ \left(\sum_{e_i \in C(e)} \left(\sigma(e_i) + \sum_{\substack{e_j \in C(e), \\ e_i \neq e_j}} |e_i| \cdot |e_j| \right) \right) \cdot RC(e) & \text{if } C(e) \neq \emptyset \end{cases}$$

⁵The length is expressed in time frames.

In addition, the extension of σ to a sequence of n edges, σ_n , is taken to be the sum of the individual scores which gives us the following inequation:

$$\sigma_2(a, b) < \sigma(c) < \sigma(d)$$

Since length is the major factor in determining the score of an edge and in distinguishing between parser results and robust rule constructions, these inherent preferences in the edge scores become more extreme for longer analyses.

The minor factors in the actual scoring function, such as reading number or argument saturation, may distinguish between parser analyses with the same span, but cannot, on their own, make a robust analysis comparable with a parser result.

Result Selection

The last step in robust processing of ill-formed input following this approach is the selection of a result, which is the input to the subsequent modules in the system. In the case of Verbmobil, the result is input mainly for the transfer module (Dorna and Emele, 1996) and for dialogue (Alexandersson et al. 1997) and context processing (Stede et al, 1998).

What counts as a result? In the optimal case, there is an edge in the VHG built up which spans the whole input. Naturally, this edge, or rather the semantic analysis annotated to it, should be output as result. If there should be more than one spanning edge, there needs to be a selection procedure which decides which edge to choose.

Often, however, it will be the case that there is no spanning edge present in the VHG. This can have several reasons. In some cases, the analysis may actually be correct, e. g. for the utterance *ja, am Montag* ('yes, on Monday'), the VIT language does not provide a way to combine the two analyses for the exclamative (*ja*) and the prepositional phrase (*am Montag*).

Another reason might be a lack of sensible partial analyses to combine, due to the input being too deficient. Or a situation might occur which is not covered by the rule set of robust semantic processing.

Regardless of the reason for the output still being fragmentary, a sequence of partial analyses has to be selected from the edges in the VHG. We will first describe the mechanism for the selection of a sequence of partial results, and afterwards that for the selection between competing spanning results. For both, we make use of the scores assigned by the scoring functions defined in 3.5.3.

Selecting a Sequence of Partial Results

If there is no spanning edge in the VHG, we have to select an optimal sequence of chart edges with respect to the scoring function σ . Effectively the problem that is posed is to produce the least segmented sequence of VITs. This can be achieved relatively easily with a standard graph search algorithm, as used, for example, to select fragment analyses from the HPSG parser's chart⁶ (Kasper et al. 1999).

Selecting a Spanning Result In a situation where we have multiple edges spanning the complete input utterance in the VHG, the selection of a result edge could also be done on the basis of the score assigned by the scoring function alone. However, the use of additional knowledge improves the reliability of the choice we make here. The knowledge employed to choose between different spanning edges comes from a statistical model of VIT sequence probabilities (Ruland et al. 1998), the so-called VIT model⁷. The VIT model was trained on transliterations of dialogues from a subset of the Verbmobil corpus.

3.6 The Rule Sets

For each of the three Verbmobil languages (German, English, and Japanese), there is a specific rule set which contains rules only applicable to analyses of that particular language. This is necessary since some kinds of information are only available for a specific language, (such as prosodic information on self-correction, which is currently only delivered for German input), and some constructions and phenomena are language-specific. Of course, there are also purely linguistic constraints on the rules that are applicable for a given language, such as word order and constituent structure. E. g., in German, an adverb may follow the verbal phrase it modifies, while an adjective has to be placed in front of the noun it modifies. If such constraints were not taken into account when combining fragments, incorrect combinations would be produced.

3.6.1 Rule Classification and Examples

The rules can be divided into four classes, cf. Figure 3.5.

⁶In practice, we actually use a standard Sicstus Prolog library predicate for selecting the path with the maximum score from a weighted graph.

⁷This model was developed in cooperation with and trained by colleagues from the University of Erlangen-Nürnberg in the context of Verbmobil.

constructive rules: Constructive rules combine two or more analyses into a larger analysis which covers the union of the interval the original analyses cover. They are monotonic in that all the information present in the original analyses is preserved in the resulting analysis. These rules basically implement the semantic composition operations which may also be performed during semantic construction.

destructive rules: Destructive rules combine two or more analyses into a larger one, but they are non-monotonic — part of the information present in input analyses is not preserved in the resulting analysis. These rules are used to model self-corrections and false starts, since those are phenomena where some part of the original information expressed needs to be removed.

type coercion: Type coercion rules map one analysis to another one spanning the same interval. They perform type coercions, e. g. mapping an analysis for a temporal nominal phrase to a corresponding temporal modifier.

format adaptations: Format adaptation rules also map an analysis to one covering the same interval. They are used to adapt the representations delivered by the parsers in cases where they are not suitable for further combination with other partial analyses.

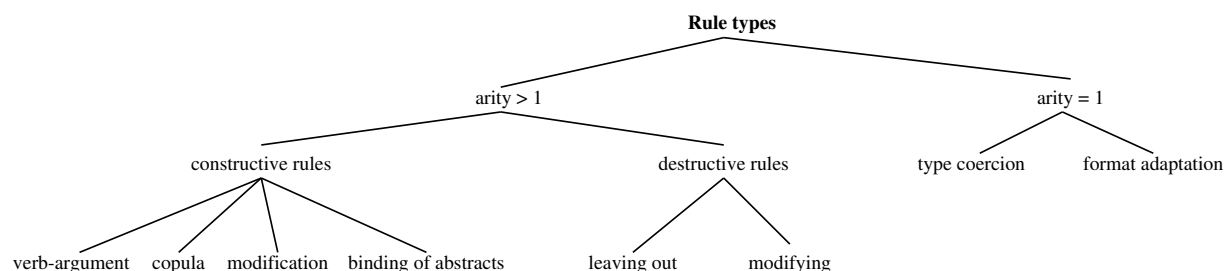


Figure 3.5: Classification of rules

The number instantiations of these types for each of the Verbmobil languages is given in Table 3.1.

Language	constructive	destructive	type coercion	format adaptation	total
German	39	10	8	5	62
English	28	5	7	3	43
Japanese	7	3	7	2	19
total	74	18	22	10	124

Table 3.1: The sizes of the rule sets for German, English, and Japanese

Constructive Rules.

Constructive rules combine two or more VITs in a monotonic way, i. e., all the information present in the VITs which are combined is still present in the resulting VIT. As an example for a constructive rule, we consider the application of a verbal functor to a potential argument.

```
(3.3) verb_arg_l ::  
  [[type(V1,term),not(real_reflexive(V1))],  
   [type(V2,verbal),missing_arg(V2),  
    possible_arg(V1,V2)]] --->  
  [apply_fun(V2,V1,V3),assign_mood(V3,V4)] & V4.
```

```
(3.4) verb_arg_r ::  
  [[type(V1,verbal),missing_arg(V1)],  
   [type(V2,term),not(real_reflexive(V2)),  
    possible_arg(V2,V1)]] --->  
  [apply_fun(V1,V2,V3),assign_mood(V3,V4)] & V4.
```

We describe the first rule in (3.3) which is one of a pair. The other one (3.4) is just its symmetric counterpart.⁸ The rule in (3.3) applies when it finds an edge annotated with a VIT which is semantically a term, i. e., it corresponds to a nominal phrase, but not a reflexive pronoun, and another edge with a VIT which is a verbal projection and missing arguments, i. e. its subcat list is not empty.

The confidence value for this functor-argument-application depends on how well the argument fits in with the requirements of the verbal functor. The computation of the confidence value takes into account the subcategorized case for the argument, the subcategorized preposition (if any), and the sortal restriction the verb imposes on its argument. The more of these restrictions that are violated, the lower is the confidence value, which influences the resulting edges score.

The scoring mechanism allows us to benefit from linguistic constraints while still having the chance to be liberal enough to produce an analysis which violates some constraints. If the verb is missing more than one argument, combination with a nominal argument will result in several new edges being entered into the VHG, one for each argument position which

⁸This pair of rules appears in the German and English rule sets, since in these two languages, verbal arguments can appear to the left and to the right of the verb. In the Japanese rule set, however, only the first rule, which combines a verb with an argument to the left, is included, since Japanese is a verb-final language.

could be filled by the nominal argument. Each of these verb-argument combinations will receive different scores, depending on which constraints have been violated. If the verb subcategorizes for an argument in nominative case and another one in accusative case, combination with a nominal argument in nominative case will result in a high confidence value and thus a high score for the edge where the argument is bound to the argument position requiring nominative case. If the argument is bound to the other argument position requiring accusative, the confidence value and edge score will be lower. Similarly, violation of sortal restrictions results in a lower confidence.⁹

This means that while the linguistic constraints are used, interpretations violating them are not ruled out completely. This increases the robustness of processing. If no analysis can be reached which satisfies all constraints, still the one which comes closest may be selected. E. g., if no combination of a verb with potential arguments fulfills all case requirements of the verb, still the most plausible combination of the verb with its arguments can be selected on the basis of the sortal restrictions the verb imposes on its arguments.

Destructive Rules.

Destructive rules also combine two or more VITs, but they are non-monotonic, i. e., some information from the original VITs is no longer present in the resulting VIT. This may occur in two ways: either some part of the information in one of the input VITs is deleted and maybe replaced, or some input VIT is not used at all when constructing the result VIT.

As an example for a destructive rule, we consider a rule which aims at filtering out a certain type of self-correction, namely those where the reparans and the reparandum result in analyses of the same type which are separated by a prosodically-marked irregular boundary.¹⁰ This is a rule which omits a complete VIT when the result is built.

The rule is shown in (3.5). It states that if an edge is found with a VIT of a certain type associated to it and which has an irregular prosodic boundary at its end, followed by another edge with a VIT of the same type, a new edge spanning the union of the two edges and associated with only the second VIT should be added to the chart. In other words, the VIT with the irregular boundary at the end is left out of the analysis.

(3.5) irreg2 ::

⁹Or, for that matter, if the verb subcategorizes for a prepositional phrase argument and we find a possible argument phrase with a different preposition (or no preposition at all), combination will still take place, but the confidence value will be decreased.

¹⁰This rule is from the German rule set, since there is no prosodic detection of irregular boundaries in English or Japanese in the current Verbmobil system.

```

[[type(V1,X), irreg_boundary(V1)], type(V2,X)]
--->
[dummy_op(0.8)] & V2.

```

This rule contains a dummy operation, `dummy_op(0.8)`, which does not perform any operation. It is only used here in order to define the confidence value for the rule. A rule without any operations would have a confidence value of 1. But since the recognition of irregular boundaries is insecure, and the rule above only expresses a heuristic, we assign it a confidence of only 0.8.

Type Coercion Rules.

Rules for type coercion (Pustejovski, 1991, Cardelli and Wagner, 1985) map an analysis to another one with a different semantic type in order to allow for a combination operation which would otherwise not be possible.

```

(3.6) tempr ::
      [type(V1,temp_np)]
      --->
      [typeraise_to_tempmod(V1,V2)] & V2.

```

The rule above takes an analysis of a temporal nominal phrase, i. e. a nominal phrase with a referent which is a temporal entity according to the sortal hierarchy, and maps it to a temporal modifier. The temporal modifier expresses that its argument stands in an underspecified temporal relation to the temporal entity.

The resulting analysis covers the same interval as the original analysis. This means that the confidence values for the type-raising operation may have to depend on whether or not that interval spans the whole of the WHG input. A lower score for a spanning analysis would ensure that if the type coerced analysis is not combined with another analysis, it will not be selected as output on its own.

Format Adaptation Rules.

The format adaptation rules again map exactly one VIT to an output VIT. In some cases, partial analyses delivered by the parser are not directly suitable for further processing by robust semantic processing, in particular for combination with other analyses. This

situation arises, for example, when the parser delivers an analysis which is partial but intended to be translatable on its own, i. e., as the spanning analysis of some input WHG¹¹. Format adaptation rules map such VIT analyses to VITs which are more suitable for further processing within robust semantics. They may be both monotonic or non-monotonic, i. e., they may remove or replace or simply add information.

An example for this situation is the analysis of topic drops in German. An utterance such as *paßt mir* ('suits me') is not complete, since the subject of the verb *passen* is missing. Nonetheless, this is an acceptable utterance in spoken German, assuming the hearer is able to resolve what the speaker refers to.

A possible analysis for such an utterance with a topic drop is to bind the subject argument position of the verb with a zero pronoun.¹² Such an analysis is useful in that the zero pronoun signals that there is a need for resolving what object in the preceding discourse the speaker refers to.

```
(3.7) zero_to_subcat ::
      [has_zero_arg(V1)]
      --->
      [zero_to_subcat(V1,V2)] & V2.
```

However, a segment that has been analyzed as a topic drop construction need not necessarily be one, particularly if it does not span the whole input. Then the subject of the sentence may be present, but it might be, e. g., separated from the verbal phrase by a mis-recognized word. In this case, the zero pronoun blocks the combination of the two partial analyses, that of the subject and that of the verbal phrase, since all the verbal argument positions are bound.

The rule in (3.7) is intended to cover such cases. It fires if it finds an analysis for a verbal projection where an argument is bound by a zero pronoun, and then creates a new analysis without the zero pronoun, where the argument position originally filled by the zero pronoun is explicitly marked as unbound on the subcategorization list. If a suitable subject is found later in processing, it can be combined with the result of applying this rule and fill the argument position occupied by the zero pronoun in the topic drop analysis.

¹¹The very fact that this is a candidate for robust operations indicates that this assumption proved false.

¹²A zero pronoun is an abstract pronoun, i. e., it is not realized on the surface. The terminology comes from the syntactic analysis of Japanese, where zero pronouns frequently occur.

3.7 Evaluation

In the course of the development of the robust semantic processing module we carried out several types of evaluation (see also Worm, 2000). The most informative of these was an end-to-end evaluation of the quality of the translations over two set of monolingual and bilingual dialogues, from the Verbmobil corpus and evaluation data, respectively. This evaluation was performed by a student of translating and interpreting with no prior knowledge of the Verbmobil system.

The quality of the translations obtained with and without the contribution of robust semantic processing were subject to a blind judgment, according to the categories: good, acceptable and bad. The result were subsequently compared and combined with the quota of missing results. The results for the translation quality on monolingual¹³ and bilingual dialogues¹⁴ are given in Tables 3.2 and 3.3.

Table 3.2: Results for translation quality with and without robust semantic processing for the monolingual dialogues

	translation			
	no result	bad	acceptable	good
without	27 (12.4%)	112 (51.4%)	49 (22.5%)	30 (13.7%)
with	37 (17.0%)	93 (42.7%)	58 (26.6%)	30 (13.7%)

Table 3.3: Results for translation quality with and without robust semantic processing for the bilingual dialogues

	translation			
	no result	bad	acceptable	good
without	23 (19.2%)	54 (45.0%)	33 (27.5%)	10 (8.3%)
with	24 (20.0%)	49 (40.8%)	36 (30.0%)	11 (9.2%)

Without the contribution of robust semantic processing, 36.2% of the turns of the monolingual dialogues received a translation which was acceptable or good, while 51.4% of the results were bad and for 12.4% of the turns no result was delivered. With robust semantic processing, the share of acceptable or good translations rose to 40.3%, which means that about 10% of the useful translation are due to the contribution of our module.

¹³g048a, g205a, g323a, g340a, and g343a from Verbmobil CDs 20 and 30.

¹⁴DDW_99_27, DDW_99_29, DDW_99_31, DDW_99_33, and DDW_99_35, a subset of the so-called ‘dialogues of the week’.

For the bilingual dialogues, the application of robust semantic processing increased the share of acceptable or good translation from 35.8% to 39.2%. This increase represents 8.7% of the useful translations.

3.8 Conclusion

In this chapter we have described a new approach to the robust processing of spoken language that is implemented within the semantics module of the Verbmobil system. Our approach has the following salient properties:

- It deals with *spoken language phenomena*, such as self-corrections or false starts.
- It deals with *ungrammaticalities*, whether caused by speech recognition errors or ungrammaticalities in the speaker's original utterance.
- It is applicable in the context of a system which processes *speech recognizer output* in the form of a word hypothesis graph.
- It exhibits the *anytime behavior* required by constraints on processing time inherent in the task of processing spoken dialogues.
- The approach has been implemented in a large coverage application and would appear to be general enough to be *scalable*.
- It makes use of *knowledge from all linguistic levels* in order to achieve good results, namely from syntax, semantics, prosody, together with heuristics which cover the cases where the hard knowledge fails to provide an answer.
- It provides a judgment about the quality and reliability of its output to facilitate the selection of a result.

Our evaluation also demonstrates that robust semantic processing makes a modest but distinguishable contribution to the overall performance of the Verbmobil system. This result is gratifying given the complexity of the system and the number of other modules that contribute to its performance. It also corresponds to the original intention of implementing a module to provide robustness against specific phenomena that are characteristic of the application domain.

Chapter 4

A Statistical Model for Predicting Dialogue Moves on the Basis of Game Structure

12

4.1 Introduction

Recognizing the dialogue act(s) performed by means of an utterance involves combining top-down expectations about the next likely ‘move’ in a dialogue with bottom-up information extracted from the speech signal. The best current models of dialogue act recognition achieve an accuracy of about 70% on transcribed words and of 65% on recognized words (Stolcke et al., 1998, Reithinger and Klesen, 1997). We are trying to improve these results by finding better ways of exploiting top-down expectations about dialogue structure.

Nagata and Marimoto(1994), Reithinger and Klesen (1997), Poesio (1991) proposed to take

¹This chapter appears as Poesio, M. and Mikheev, A., "The Predictive Power of Game Structure in Dialogue Act Recognition: Experimental Results Using Maximum Entropy Estimation", *Proceedings of ICSLP 98*

²We wish to thank the other members of the TRINDI consortium, in particular Robin Cooper and David Traum; we also want to thank Jan Alexandersson, James Allen, Matthew Aylett, Chris Brew, Mark Core, Steve Isard, Daniel Jurafsky, David McKelvie, Steve Pulman, and Norbert Reithinger for comments and discussions, as well as audiences at the 1997 NGO workshop on Robust Processing of Dialogue in Nijmegen, at the HCRC's Dialogue Working Group, and at the University of Sheffield's Natural Language Group.

advantage of expectations about what comes next in a conversation in order to improve dialogue act recognition. Statistical techniques to acquire this information were proposed by Nagata and Marimoto (1994) and Reithinger and Klesen (1997) among others. These researchers noted the resemblance of this task to that of part-of-speech tagging, and applied therefore techniques developed for this latter purpose, such as n-grams and hidden Markov models (Bahl et al., 1983, Katz, 1987), making the prediction of the next dialogue act depend on the previous n dialogue acts. Reithinger (1995) examined the predictive power of these models in isolation from bottom-up information, and found that they could achieve an accuracy of about 39%; he also noticed that bigrams performed as well as higher-order models (this result was confirmed in Stolcke et al., 1998). We likewise concentrated on the predictive power of expectations in isolation.

4.2 Game-Based Expectations

It has been suggested in conversation analysis (Levinson, 1983) that both ‘local’ and ‘global’ organizing principles are at play in spoken conversations; these elements of organization include so-called ADJACENCY PAIRS (such as, e.g., QUESTION-ANSWER or INSTRUCT-ACCEPT) at the local level, as well as global organization principles such as, for example, the fact that phone conversations are normally opened by a signal from the person receiving the call, followed by the called introducing herself and the reason for the call, etc. It was also found that while these organization principles are not always respected, deviations from the norm tend to be marked—e.g., by pauses before rejecting a request. These findings led researchers to hypothesize that the participants in a conversation tend to act in accord with conventional routines called ‘scripts’ or GAMES (Carlson, 1983, Levin and Moore, 1978, Houghton and Isard, 1987) that generate expectations about what is coming next.

Although n -gram models can be viewed as a stripped-down version of game-based expectations, some information is lost when conversations are seen as simple sequences of moves - namely, the difference between *intra*-game and *inter*-game predictions. In a QUESTION-ANSWER game, for example, the initial QUESTION may be followed by a CLARIFY subgame. When this ends, the conversational participants return to the original question, and there is therefore an expectation that an ANSWER will follow—i.e., it is as if the subgame had not been there. (See Fig. 4.1.)

Also, whereas the (*intra*-game) prediction that a QUESTION will be followed by an ANSWER or a REQUEST FOR CLARIFICATION is fairly reliable, predicting what move will follow a move that closes a game is much harder, and if at all, it depends on even higher levels of structure. (E.g., the TRAINS conversations (Gross et al, 1993, Heeman and Allen, 1993) follow a rather predictable pattern whereby an initial phase in which the users ask various questions of the system, they then start making suggestions.)