Using the Framework

The FRACAS Consortium

Robin Cooper, Dick Crouch, Jan van Eijck,
Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp,
David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman

With additional contributions from:
Ted Briscoe, Holger Maier and Karsten Konrad

FraCaS

A Framework for Computational Semantics
LRE 62-051

Deliverable D16

January 1996

LRE 62-051 FRACAS

A Framework for Computational Semantics

CWI Amsterdam

University of Edinburgh
Centre for Cognitive Science and Human Communication Research Centre

Universität des Saarlandes
Department of Computational Linguistics

Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung

SRI Cambridge

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

The FRACAS Project Administrator
University of Edinburgh
Centre for Cognitive Science
2 Buccleuch Place
Edinburgh EH8 9LW, Scotland, UK
`fracas@cogsci.ed.ac.uk`

Copies of reports and other material can also be accessed via anonymous ftp from `ftp.cogsci.ed.ac.uk`, directory `pub/FRACAS`.

Contents

Using the Framework

The FRACAS Consortium

Robin Cooper, Dick Crouch, Jan van Eijck,
Chris Fox, Josef van Genabith, Jan Jaspars, Hans Kamp,
David Milward, Manfred Pinkal, Massimo Poesio, Steve Pulman

With additional contributions from:
Ted Briscoe, Holger Maier and Karsten Konrad

Abstract

There are two major levels of processing that are significant in the use of a computational semantic framework: semantic composition for the construction of meanings; and inference either for the exploitation of those meanings, or to assist in determining contextually sensitive aspects of meaning. The first chapter of this deliverable outlines the semantic competences that are either required by — or would improve the performance of — a variety of different commercially relevant applications. Semantic composition and meaning construction is a core competence required by all applications, and inference is central to many. The second chapter takes the form of a manual describing the use of a program library and educational/research tool providing a concrete computational framework bringing together different types of semantic composition. The third chapter presents an inference test suite for evaluating the inferential competence of different NLP systems and semantic theories. Providing an implementation of the inference level was beyond the scope of FraCaS, but the test suite provides the basis of a useful and theory/system-independent semantic tool. The fourth chapter gives an overview of the state-of-the-art in computational lexical semantics. A considerable amount of time on the project was devoted to discussing lexical semantics, and revealed that the computational and formal semantic traditions are not as far apart as might first be assumed from the literature. Lexical semantics is a major semantic competence, and has an impact both on semantic composition and on inference. A deeper understanding of lexical issues is required even to separate out in a useful way those inferences in the test suite that hold for structural reasons and those that hold for lexical reasons. The final chapter discusses significant themes on computational semantics, and further directions for extending the frameworks

developed during FraCaS. At present this chapter is being circulated in draft form to the project reviewers, and will be included in the final version of this deliverable after it is discussed at the FraCaS final review.

Chapter 1

The Relevance of Semantics for Commercial Applications

It is difficult to say very much of detail about the usefulness of semantics for practical applications, because the state of the art is such that there are no commercial applications of any type which involve a substantial semantic component. There are even very few research prototypes with any degree of semantic coverage. Thus the following paragraphs are more in the way of informed guesswork rather than hard fact: we attempt to list some of the main applications of NLP and say to what extent (if at all) semantics is already involved, and whether performance might be improved if it were.

To begin with, it is useful to distinguish three types of semantic processing that a system might undertake. Firstly, we distinguish lexical semantic processing, by which we mean processing that involves the meanings of words in some way. This might be simply by distinguishing between different senses of words, or positioning words in a semantic domain or hierarchy via a thesaurus, through to associating words with meaning postulates in order to be able to describe various complex lexically-based inferences etc. The second type of processing we will describe as 'structural semantics', by which we mean the process of recovering whatever information we can about the context-independent meaning of a sentence, on the basis of the words in it, their meanings, syntactic structure, and whatever semantic operations or properties may be detected on a purely linguistic or structural basis. The third type of processing we describe as 'contextual processing', namely the recovery of whatever context-dependent elements of meaning are necessary to understand the sentence.

Clearly there is a logical dependency between these types of processing, and equally clearly they make increasingly greater demands on system builders. Contextual processing in particular involves non-linguistic reasoning of various sorts, and as has been stressed in other FraCaS documents, currently represents one of the hardest research challenges in NLP.

We now briefly list some typical current and future NLP applications, and say which types of semantic processing they already use (in at least some instances), which types they could use with profit, and which types of semantic processing are, in our view, absolutely necessary if satisfactory performance is ever to be achieved.

1. Grammar correction and spell checking: This might be in the context of a simple editor or word-processing package, or in the context of a document composition system which was attempting to enforce a 'house style' on the writer. There are probably some cost-benefit calculations to perform before adding semantic processing of any kind to such systems, but it is likely that detection of unintended ambiguities, and some kinds of error, could be performed more accurately with some lexical and structural semantic processing.

2. Structured document composition: By this we mean the next stage of sophistication from the preceding kind of system: for example, a tool for writers producing technical documentation, system specifications, or even legal documents. The system would carry out consistency checks on terminology, detect possible new technical terms, enforce the use of definitions at appropriate points, and in general be aware of the intended structure and form of the document. Detection of possible technical terms can be done on a purely syntactic basis, but is more accurate if some lexical semantics can also be used. Keeping track of definitions requires being able to relate different syntactic forms to the same concept.

3. Information extraction: This includes simple applications like message routing, through to template or form-filling from free text like newspaper or intelligence reports. Current systems usually work in domains too restricted for lexical ambiguity to be a problem, and most things that look 'semantic' are actually done by (hand-crafted) pattern matching. But accuracy, portability, and the ability to cope with different domains simultaneously, can be better achieved by doing at least lexical and structural semantics. It is also the experience of many participants in the MUC projects that contextual processing, especially anaphora resolution, improves performance considerably in some aspects of the task, even where the contextual processing itself might be done using rather minimal semantic apparatus.

4. Interactive translation and machine aided translation: This is intended to include systems which involve a dialogue (perhaps a very restrictive one) between a person and a machine (to arrive at a translation) or between two people (to solve a problem). The key point is that there is the possibility to fall back on a user to supply information needed to make translation decisions. Like all translation, interactive translation requires disambiguation or translation decisions that are based on lexical or structural semantic factors. Contextual processing is not absolutely necessary if translation proceeds on a sentence by sentence basis, for there is always the human agent to fall back on. But if it is available and sufficiently accurate, interactions are less complex and more natural. If the interactive translation is within a dialogue context then clearly at least some degree of contextual processing is usually necessary.

5. Offline translation: Usually large scale translation of texts, not involving human intervention (at least, not in the part of the process we are interested in: there may be pre- or post-editing involved in the overall system). As is well known, high quality translation between many language pairs (e.g. Japanese to English, although less true in the reverse direction) can only be achieved if contextually determined properties like (zero) anaphoric reference and definiteness can be satisfactorily resolved.

6. Interfaces to information systems: E.g. database query; front ends to advice giving or order processing systems; control of equipment. We assume that these will often be spoken language systems. Interfaces are generally intended to support dialogues and anaphoric or other reference to previous utterances or system output arise quite naturally and must be supported. Furthermore, relating linguistic content to e.g. a database model requires reasoning on the basis of a fully articulated representation of the meaning of the utterance.

7. Text generation: Systems that turn non-linguistic representations into sentences. Examples would be financial report generators; those that report on the running status of some piece of equipment; etc. Text generation almost by definition involves taking some meaning representation (whether linguistic or otherwise) and producing sentences with the same meaning. Thus at the very least an explicit lexical and structural semantic model must be employed, as well as syntax and morphology. If whole texts are being produced then sophisticated contextual processing is required to ensure that they are coherent and natural, rather than just sequences of independent sentences.

8. Text to speech: Speech synthesis either from free text or structured linguistic input. Properties like information structure (given/new, focus, etc.) are major determinants of intonation. To be able to produce fully natural sounding speech, a text to speech system should be able to determine these properties, although as yet only a few small scale experimental systems do so. In general, you need to understand the meaning of a sentence if you are to be able to speak it naturally: this is true just as much for machines as for humans.

In the following table we summarise these remarks about the current state of play with respect to whether at least some (usually not all) existing systems already use lexical (L), structural (S), or contextual (C) semantic processing. In the third column, we indicate which levels of processing would if also carried out improve performance (we assume) in these systems. (Of course, we assume that *improved* rather than additional semantic processing would always improve task performance.) In the last column we indicate whether (in our view) a system performing this task can ever achieve a satisfactory level of performance if it does not carry out this kind of semantic processing.

Application	already uses	could use	must use
1. Grammar correction		L S	
2. Document composition		L S C	
3. Information extraction	L S	C	L
4. Interactive translation	L S	C	L S
5. Offline translation	L S	C	L S
6. Information interfaces	L S C		L S C
7. Text generation	L S C		L S C
8. Text to speech		L S C	

Chapter 2

The Framework Tools

2.1 The CLEARS User Manual

2.1.1 Introduction

The *CLEARS* system (Computational Semantics Tool for Education and Research in Semantics) was developed as part of the FraCaS (Framework for Computational Semantics) project which is designed to encourage convergence between different semantic formalisms.[1] There are currently a wide variety of semantic formalisms used in teaching and research, for example, various versions of 'Montague-Grammar', DRT, Situation Semantics and Property Theory. As different as they look on first sight, they share many common assumptions, and provide similar treatments of many phenomena. The tool allows exploration and comparison of these different formalisms, enabling the user to get an idea of the range of possibilities of semantic construction. It is intended to be used as both a research tool and a tutorial tool. In this paper we describe how to use the system. The manual is divided into four parts. The first part of the paper serves as a quick start into *CLEARS*. The second part is the mere description of the features of the system. After that we show with selected examples how to use some the system in a framework of teaching or research. The last parts consists of appendices, that give a quick overview over commands and provide other useful information.

Quick Start Section 2.1.2 describes how to obtain and install the system, the prerequisites and — most important — how to start the system, which you can see in Figure 2.1. In section 2.1.3 we give a worked example

[1]This section is a preliminary version of a user manual to appear in [Maier *et al.*, 1996].

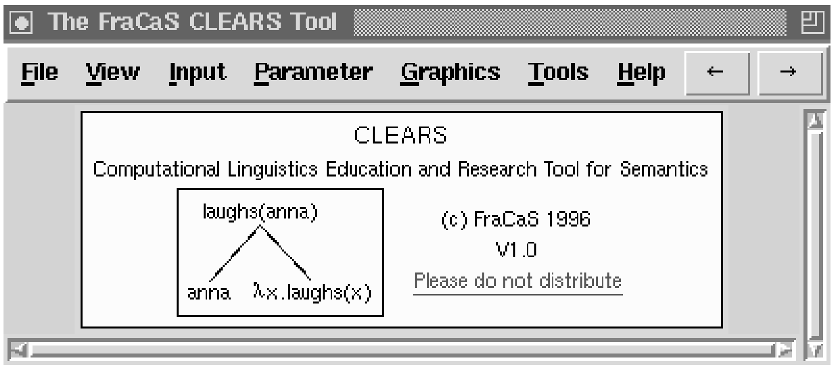


Figure 2.1: The User Interface of CLEARS

of how to use the system and explain the motivation of the system as a teaching tool.

Description of the System In 2.1.4 we give an overview over the pull-down menus of *CLEARS* and describe shortly their usage. Then, in 2.1.5 we do the same for the pop-up menus that are associated with parts of the graphic output and that provide the basic for the interactive use of the tool

Selected Topics in using ***CLEARS*** Here we present some ot the possibilities of using *CLEARS* as a system for teaching or research.

Appendices In the last part of the paper we provide a short reference guide. We present overviews concerning the most important commands and tables that summarise the parameters.

2.1.2 Installation

How to obtain CLEARS

You can anonymously ftp the system

```
ftp ftp.coli.uni-sb.de
> cd /pub/Fracas
> bin
> get clears.tar.gz
```

Alternatively you can get it, from the *CLEARS* homepage `http://www.coli.uni-sb.de/~fracas/` using your favourite web-browser.

To install the system you need

System	Developer's Web Address	Freely Available
Sicstus Prolog 3	Swedish Institute for Computer Science `http://sics.se`	no
Tcl 7.4 Patch Level 0	J. Ousterhout `http://www.smli.com/people/john.ousterhout/`	yes
Tk 4.0 Path Level 0	J. Ousterhout see above	yes

To our knowledge, the SICStus Tcl/Tk Interface unfortunately works only with Patch Level 0 of the latest Tcl/Tk distribution. We hope that this will be fixed very soon.

You can install the system in the directory of your choice. However, we propose to create a directory named Fracas. For the rest of this manual, we use $FRACAS as the placeholder for your installation directory and $LIBRARY as a placeholder for the directory in which your local system provides linkable libraries[2].

```
> mv clears.tar.gz $FRACAS
> gunzip clears.tar.gz
> tar -xvf clears.tar
```

This will create the appropriate subdirectories. You have to set two new environment values[3] to use the program. We recommend to modify your .cshrc file, so that these variables are set permanently.

```
> setenv CLEARS_DIR $FRACAS
> setenv CLIG_DIR $FRACAS/Clig
```

The Tcl/Tk Environment

Before you start you have to make sure, that Tcl/Tk is installed at your site and you have set or modify the following environment variables to enable SICStus Prolog to find the appropriate libraries.

[2]If in doubt ask your superuser. Usually this directory should be `/usr/local/lib`.

[3]Shell Commands follow the `csh` syntax. If you run another shell, please check the corresponding commands in your manual.

```
    setenv LD_LIBRARY_PATH
    ...:$LIBRARY/tcl7.4:$LIBRARY/tk4.0
    setenv TCL_LIBRARY $LIBRARY/tcl7.4
    setenv TK_LIBRARY $LIBRARY/tk4.0
```

Tcl/Tk is an interpreted Script language, that eases the pain of creating user-interfaces in X-Windows. The system, developed by John Ousterhout [Ousterhout, 1994], is freely available and seems to emerge as a standard for rapid prototyping of GUI-based systems. If you don't have Tcl/Tk, you can get it via the World Wide Web from the page `http://www.sunlabs.com/research/tcl/`. You will also find lots of useful information concerning these packages. You can also ftp the system from `ftp.aud.alcatel.com` which is the primary site for the Tcl/Tk archive.

Running Clears

To find your local version of SICStus Prolog , you should ask your system administrator. Make sure you are in the $FRACAS directory.

Usually, you should be able to launch SICStus Prolog with the command

```
> sicstus
```

Now you can start the *CLEARS* system with either consulting or compiling the top-level file make.pl, which loads the files of this distribution. This means that at the Prolog prompt you should type in either one of these commands:

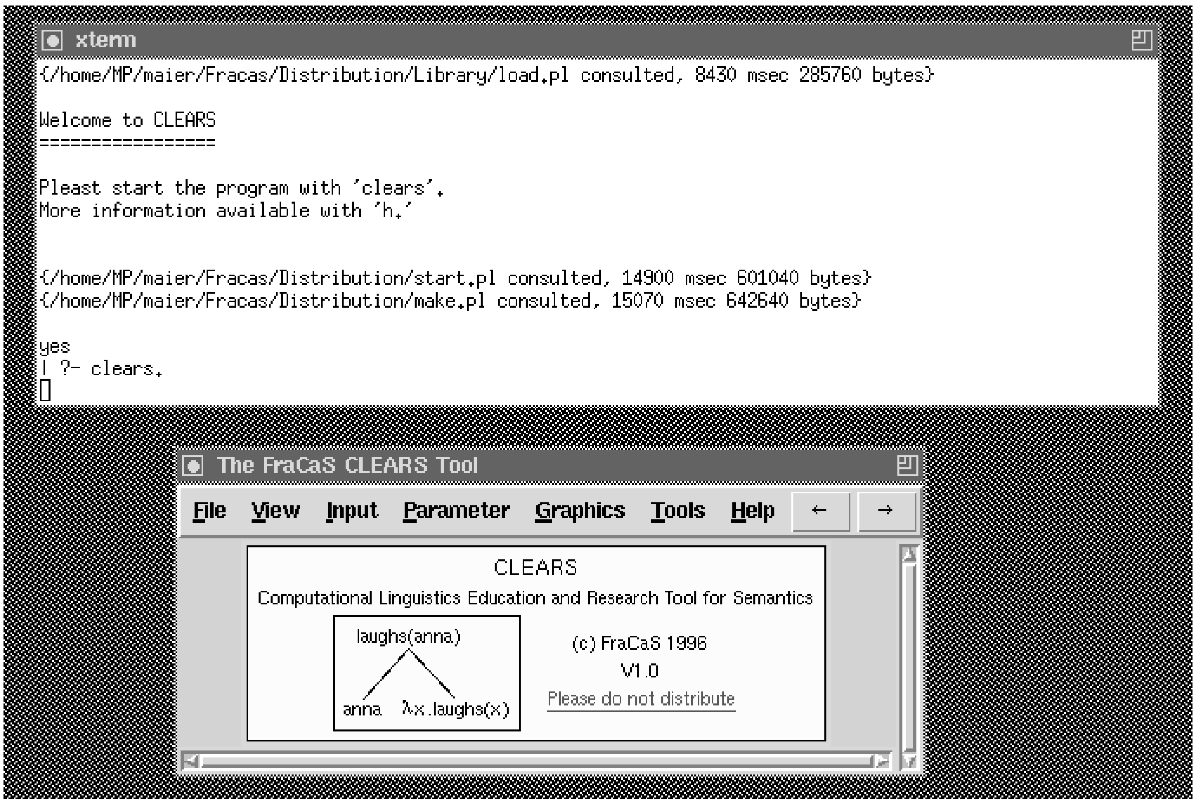
```
    | ?- compile(make).
    | ?- consult(make).
```

Compilation makes the Prolog parts of this distribution (and only these) about 8 times faster than consulted code, according to the SICStus manual [SICStus, 1995].

After loading the system you should see this welcome message:

```
Welcome to CLEARS
=================

Please start the program with 'clears'.
More information available with 'h.'
```



```
{/home/MP/maier/Fracas/Distribution/Library/load.pl consulted, 8430 msec 285760 bytes}

Welcome to CLEARS
=================

Pleast start the program with 'clears'.
More information available with 'h.'


{/home/MP/maier/Fracas/Distribution/start.pl consulted, 14900 msec 601040 bytes}
{/home/MP/maier/Fracas/Distribution/make.pl consulted, 15070 msec 642640 bytes}

yes
| ?- clears.
```

Figure 2.2: The FraCaS CLEARS Tool

Now you should start the system, with

```
| ?- clears.
```

This should invoke the graphical user interface of CLEARS. Your screen then should look similar to the on in Figure 2.2.

Trouble Shooting

You may come across some behaviour of the system different from the description above. In most cases this will only indicate that you didn't set your environment variables correctly.

No Environment Variables If the system finds no value for the variable `CLEARS_DIR`, it will start the following dialog:

```
ERROR while loading CLEARS:
Could not read the Environment Variable CLEARS\_DIR.

You have two possibilities:

(1)  Go on and let the system set this variable
     TEMPORARILY to your actual directory

(2)  Exit and set the variable either in your shell or your .cshrc File

Please enter your selection

If you choose the first option
```

However it is very likely[4] that in this case you also did forget to set the CLIG_DIR variable. But then, trying to start *CLEARS* will result in the following error message:

```
{TCL ERROR: tcl\_eval/3 - couldn't read file
"./graphbox.tcl": No such file or directory}
```

Please exit SICStus Prolog with `halt.` and either set these environment variables permanently in your `.cshrc` File as shown in Section 2.1.7 or start the shell script `clears`, which is described in Section 2.1.2.

Tcl/Tk Library not found A much more serious error is

```
{EXISTENCE ERROR: tcl\_new(\_35): procedure tcltk:(tcl\_new/1) does not exist}
```

after you tried to start *CLEARS* in Prolog. In this case, you may also have noticed, that during the loading of the library `tcltk.ql` a fatal link error occurred, like:

```
library -ltcl: not found ld: fatal: File processing errors.
```

During the startup process, SICStus Prolog could not find the libraries for Tcl/Tk. Consult your system administrator and check the default setting for the TCLLIB variable in the SICStus Makefile. You should check the exact location and name of the libraries and what setting of the TCLLIB was used during installation!

[4] In case you did not already install the *Clig* system

The Shellscript clears

This simple shell script for the c-shell will automatically set the appropriate environment variables, starts SICStus Prolog and *compiles* the system. Again you have to issue `clears` at the Prolog prompt to start the system.

2.1.3 Semantic Composition at your fingertips

In this section, we give two worked example of possible interactive sessions with *CLEARS* . They should give a first impression of how to use the system in a teaching environment. In our examples we mainly focus on how to obtain a semantic representation for an input sentence.

A first example

Our first example will show step by step how to get a semantic representation for the sentence `anna laughs`. To represent the meaning of each word and each phrase in the derivation, we choose the *Language of Generalised Quantifiers* as a representation language. This is an extensional fragment of a formalism in a Montagovian spirit.

Assuming you successfully started the *CLEARS* system, the first step is always to choose the semantic formalism with which you want to work with.
Therefore you have to click on the **Parameter** Menu from which you choose the first entry **Semantics**. Clicking on that menu opens a cascaded menu with all possible semantic representation formalisms available in *CLEARS* . You should choose the entry **Gen. Quantifier**. These two menus are displayed in Figure 2.3. Also for the next parameter to select click on the **Parameter** menu. Choose the entry **Subj-VP Application** which also opens a cascaded menu with two entries, from which you select **Apply(VP,Subj)**.

Now click on the **Input** Menu and choose from its pull-down menu the entry **Sentence**. An editor window will appear on your screen, waiting for your input. As our example sentence is `anna laughs`, you should activate the entrance field[5]. and enter our example sentence. This already has been done in Figure 2.4. With clicking on the **Send To CLEARS** button in the rightmost corner of our edit window, we send the input to the system for evaluation. *CLEARS* parses the sentence and draws a parse tree in its main output window, as Figure 2.5 shows. You will notice that the tree is annotated not only with the syntactic categories, but also with semantic operations at each non-terminal.[6] But let

[5]Activation is done by clicking with the left mouse button on that field.

[6]The preterminal nodes are not annotated with semantic interpretation. Because the semantic information of a preterminal is always identical to the lexical semantics of its (terminal)

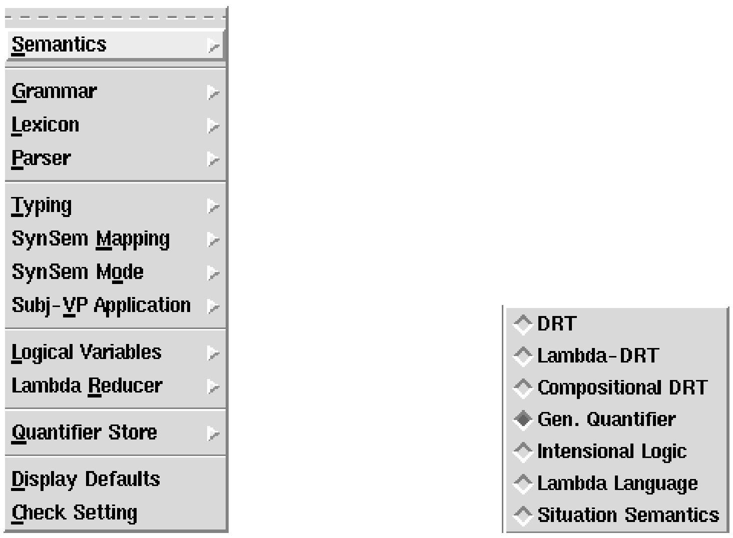
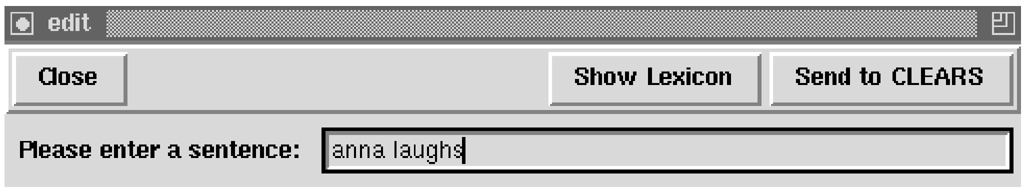


Figure 2.3: Choosing the semantic formalism



Please enter a sentence: anna laughs

Figure 2.4: The Sentence Editor of CLEARS

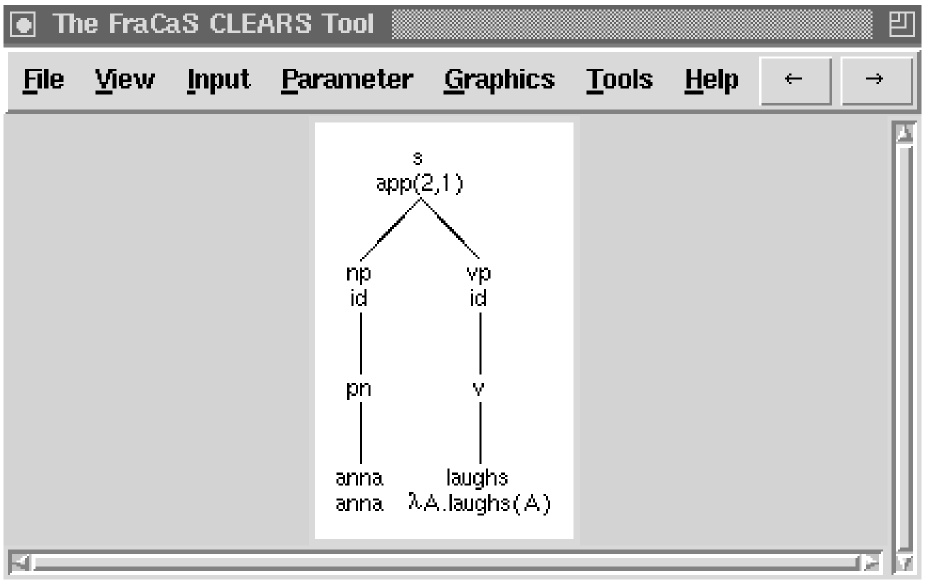


Figure 2.5: Syntax Tree of the Input Sentence

us first take a look at the terminals, the lexical information. Each word of our input sentence is connected to its lexical semantics. From this nuclear semantic information we compose the meaning of the whole sentence. So in this section, you will see how Frege's *Principle of Compositionality* [Frege, 1879] comes to life with a click of your mouse. For this purpose the semantic operations at each non-terminal node[7] tell us how the meaning of that node is composed from the meaning of its children. So the meaning of the `np` node is the result of applying **Id** — which stands for Identity — to its only child. In this case this means percolation of the lexical semantics of `anna`, which is **anna**[8] to the `np` node. Move your mouse to the **Id**-Command connected with the `np`-node and double-click[9] with your right mouse button on it. Your *CLEARS* window should now look like Figure 2.6. Please derive now the meaning of the `vp` node by clicking on its semantic node. The result is shown in Figure 2.7. Now we will derive the meaning of the whole sentence. The operation connected with the `s` node is **app(2,1)**. Clicking on this node means applying the semantic information of the second daughter to the semantic information of the first daughter. In general, when clicking on an **app** operation, *CLEARS* fetches the

child, we choose to skip semantic information at that level, otherwise this would only involve unnecessary copying. We propose to consider the preterminal together with its child as a single unit.

[7]Again, please exclude preterminals from your consideration.

[8]For convenience, we use the style `Typewriter` for syntactic information and **Bold Face** for semantic information.

[9]You will notice that while moving your mouse over the graphic output, the curser changes sometimes from an arrow to a hand. The hand indicates that these areas are clickable. They are connected with commands that drive the composition process or that manipulate the output.

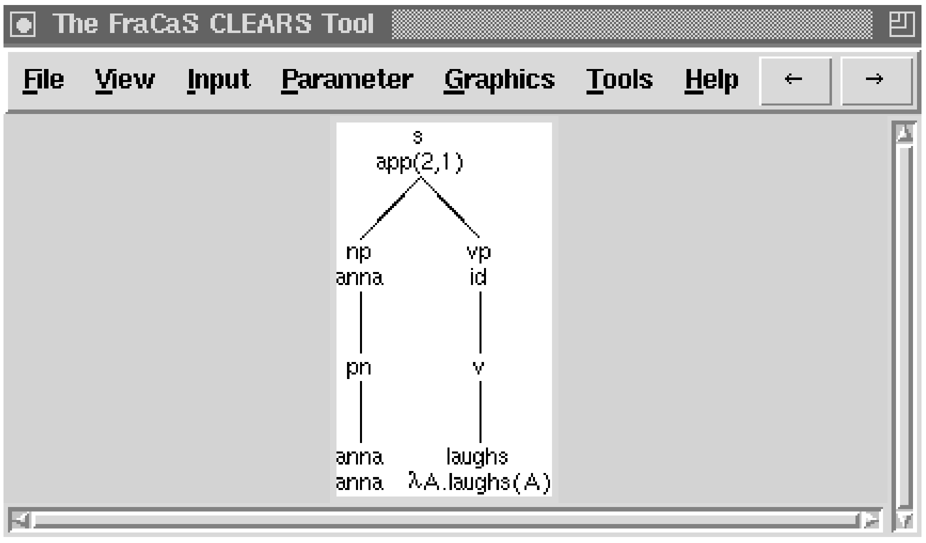


Figure 2.6: Constructing the Semantics of `NP`

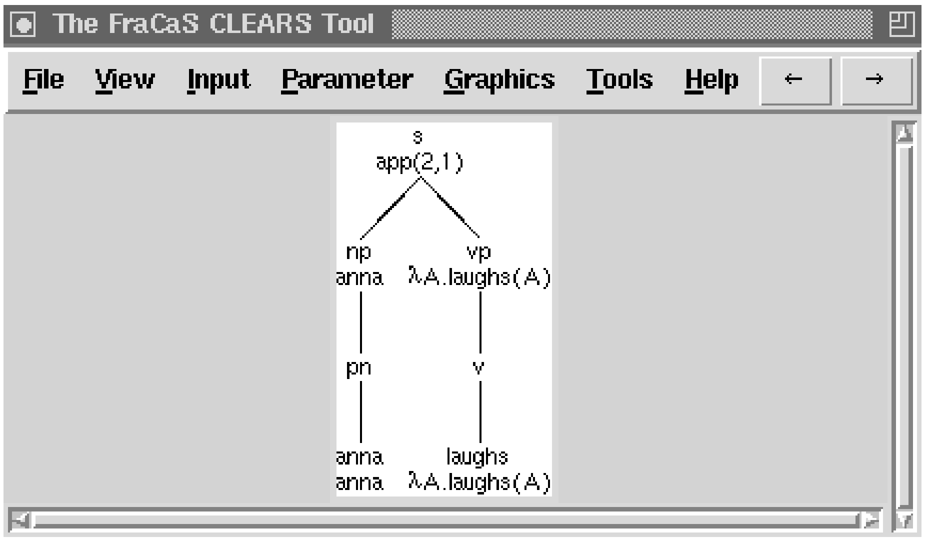


Figure 2.7: Constructing the Semantics of `VP`

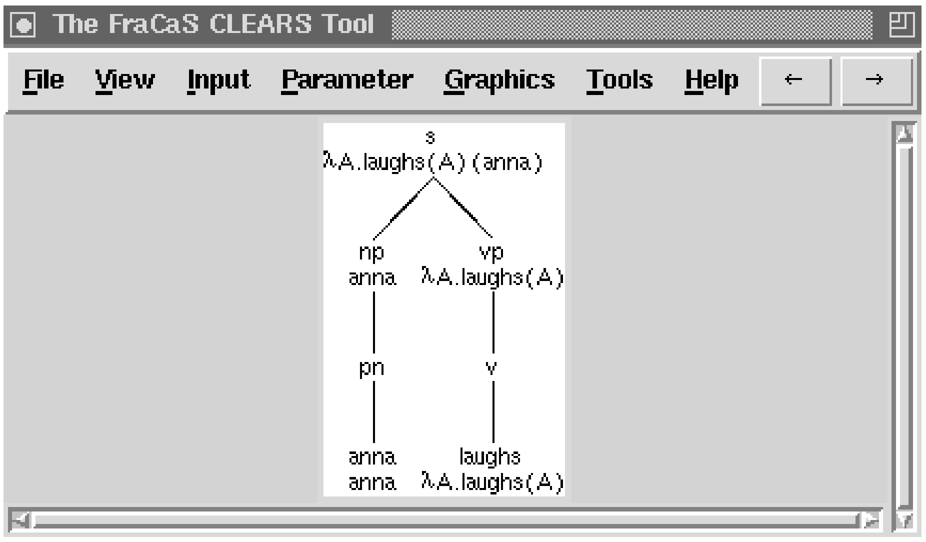


Figure 2.8: Construction of an Application

semantic information of the daughters and constructs an application according to the order specified through the arguments of **app**. In our case, *CLEARS* takes **anna** and λ **A. laughs(A)** and combines them to the application

λ **A. laughs(A) (anna)**

as Figure 2.8 shows. Now the final result is only one click away. We can reduce the semantic representation at the `s` node using β-reduction. That means a term of the form $(\lambda x.M)N$ can be reduced to $[N/x]M$, that is substituting all free occurrences of x in M with N. The result (which — surprise, surprise — you achieve through clicking on the application) is shown in Figure 2.9. Congratulations, you interactively constructed your first semantic representation. Now you may want to see all the intermediate stages to recapitulate this process. Well, click on the **Graphics** menu and select the checkbutton **Draw Stack**. Click again on **Graphics** and select also the checkbutton **Draw Boxes**. Again click on **Graphics** and select the last entry **Change Graphics**. The result in Figure 2.10 shows all steps of your derivation of the meaning of `anna laughs` *stacked* on top of each other and separated by *boxes*.

A more elaborate example

After this introductory example, we present how to obtain the two different readings of the sentence `every man loves a woman` using λ-DRT as our representational formalism. You will see how to use *Cooper Storage* as a device for

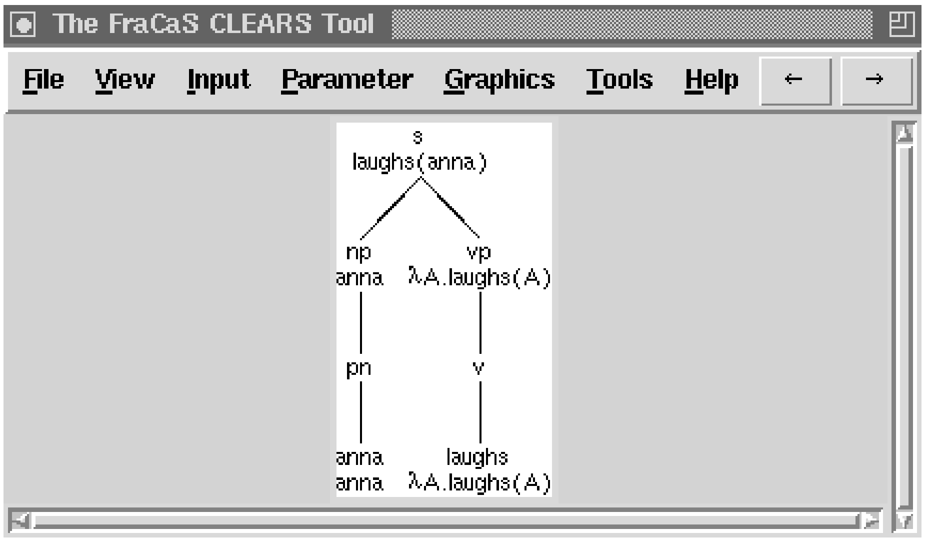


Figure 2.9: Final representation of `Anna laughs`

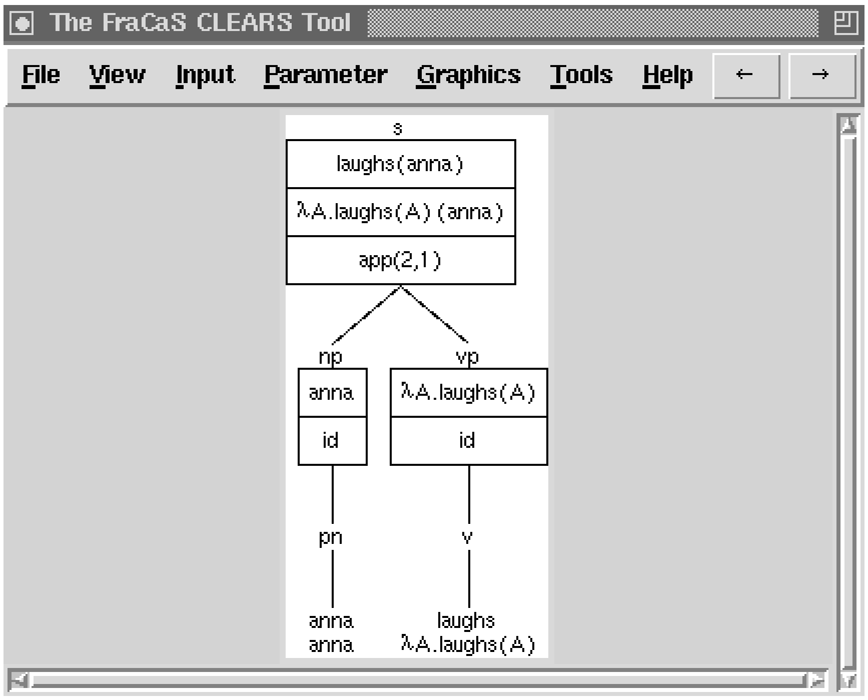


Figure 2.10: Seeing the whole construction process

obtaining the different readings.

To describe the parameter settings, we will print a Path showing the names of the menu entries which you have to click on. So **Parameters/ Semantics/Lambda-DRT** means that you have to click on the top-level menu **Parameters**, choose the entry **Semantics** and click on the radiobutton for **Lambda-DRT** in the cascaded menu that appears.

Our parameter settings in this new format are

> **Parameters/Semantics/Lambda-DRT**
> **Parameters/Subj-VP Application/Apply(Subj,VP)**
> **Parameters/Quantifier Store/Cooper Storage**

Note that we did not change all possible parameters, but the necessary ones to obtain the correct result. You could also try to use Grammar 1 and[10] Lexicon 1. However we recommend to delay the experiments with the parameters until you know more about them. An inconsistent parameter setting may cause subtle problems that are not easy to spot.
After changing the parameter, please enter the new sentence `every man loves a woman` into the edit window, which you, as you already know, get under **Input/Sentence**.
After having sent the input to *CLEARS* , you should see a screen similar to Figure 2.11. Now double-click with the right mouse button on the **Id** operations below the two `nbar` nodes. After that move your mouse to the **app(1,2)** node of the first `np` and double-click on that node. This will result in constructing an application following the procedure outlined in the previous example. The next double-click will save the semantic representation of the `np` node in the storage, later to be discharged at the `s` node. The store consists of a sequence of variables, each of it is connected[11] with a formula that can be discharged. Graphically the store is separated from the remaining formula by a surrounding box. it, as you see in Figure 2.12. Now you should double-click with the right mouse button 2 times at the **app(1,2)** operation associated with the second `np` node. After that, repeat two double-clicks at the semantic operations below the `vp` and the `s` node[12]. At the `s` node you should now see the store, consisting of two entries, that means two variables and their associated formulas. Next to the store, you see the stripped semantic representation of our input sentence. The variables in the argument of this representation correspond with the dischargeable variables[13] in the store.

[10]The conjunction is absolutely necessary. If you choose to change the grammar to **Grammar 1**, you have to change your lexicon to **Lexicon 1**, too.
[11]This connection is displayed by the use of the `::` operator between variable and formula
[12]And please do it in that order.
[13]The variables in the store with additional boxes around them.

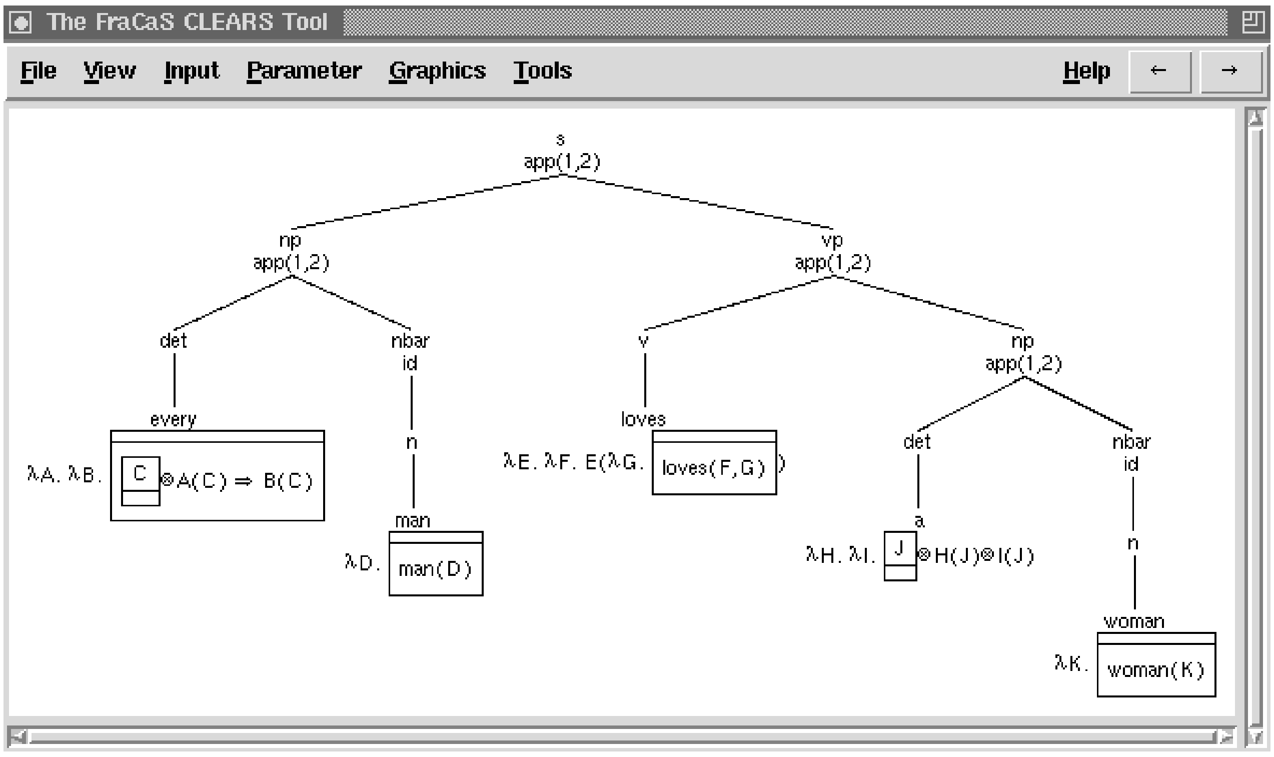


Figure 2.11: Initial Representation of `every man loves a woman`

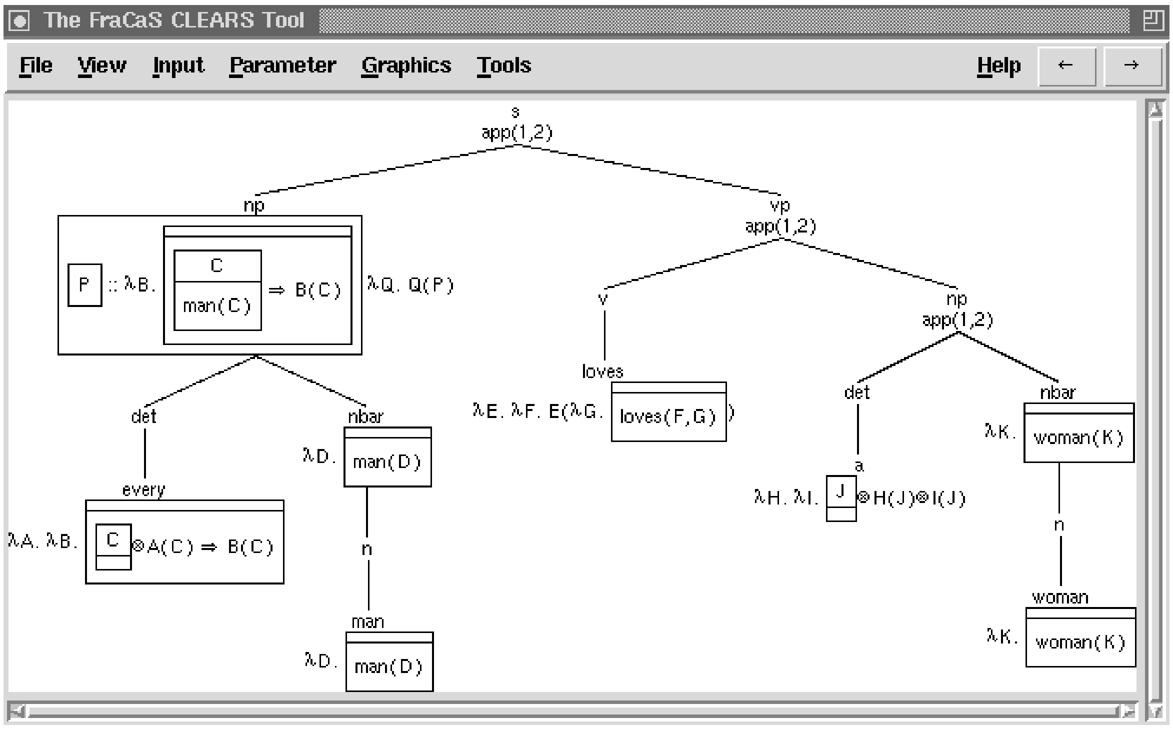


Figure 2.12: Representation of a Quantifier Store

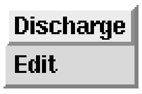


Figure 2.13: The PopUp menu associated with dischargeable variables

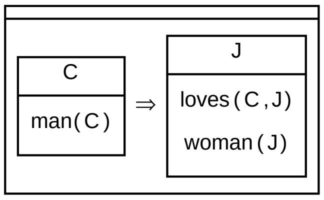


Figure 2.14: Final representation of one reading of our input sentence

To obtain a reading you have to discharge the variables in the store. Move your mouse to the store, to the variable R associated with the denotation of `a woman`. Click on the variable without releasing the button and you will see a pop-up menu, like that in Figure 2.13 appearing on the screen, from which you should choose the **Discharge** entry. This will remove this variable from the store, substituting its occurrence in the stripped semantics with the formula it is connected to. Now discharge the other variable and you get a representation without the store, but with the second DRS not fully merged. Double-click again on this semantic node to obtain the final representation. This final semantic representation of our input without drawing the whole tree, is shown in Figure 2.14. If you choose a different order of discharging your variables from the store, you get the other reading of that sentence, which you can see in Figure 2.15. This completes our guided tour through *CLEARS*. We saw how to choose parameters from pull-down menus and you saw how to manipulate a structure through a pop-up menu. Pop-up menus that contain commands providing the interactive behaviour of the system are connected to every class of structures of our parse tree. Usually on the left button you have commands

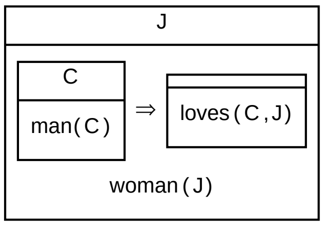


Figure 2.15: Representation of the second reading of our input sentence

that are useful in further analysis of the structure.On the middle button you find menus containing commands that allow to edit or manipulate the graphical output. The right button is usually associated with a single command, that should be the most frequently used command for this class of structures.

Especially, on the popup menu of semantic nodes invoked with the left button of the mouse, you find different commands for semantic construction. During the construction process you may choose between those commands to construct the next representation. In Figure 2.16 we show schematically how to obtain a semantic representation of an input sentence.

2.1.4 Pull-Down Menus

The *CLEARS* system automatically installs itself on top of the interactive grapher *CLiG* . A detailed description of the *CLiG* system is given in D15. Here we will just explain the specific menu bars and entries of *CLEARS* . These are the Input, Parameter, Graphics and Tools Menu, that you already have seen in Figure 2.1.

The Input Menu

Here you choose your basic input, either a sentence or an individual semantic or syntactic representation that you want to have displayed. The menu contains the following entries:

Sentence lets you enter a single sentence to be analysed.

Discourse enables you to enter a discourse sentence by sentence. In the present version of *CLEARS*, this only works for DRT.

Semantic Representation allows you to enter a single semantic expression in our Prolog-representation

Syntactic Representation allows you to enter a syntax tree.

Show Lexicon displays the actual lexicon according to the parameter setting. This is especially useful, if you don't know the coverage of the lexicon.

Show History displays all your input connected with the parameter setting and lets you choose one of your former results.

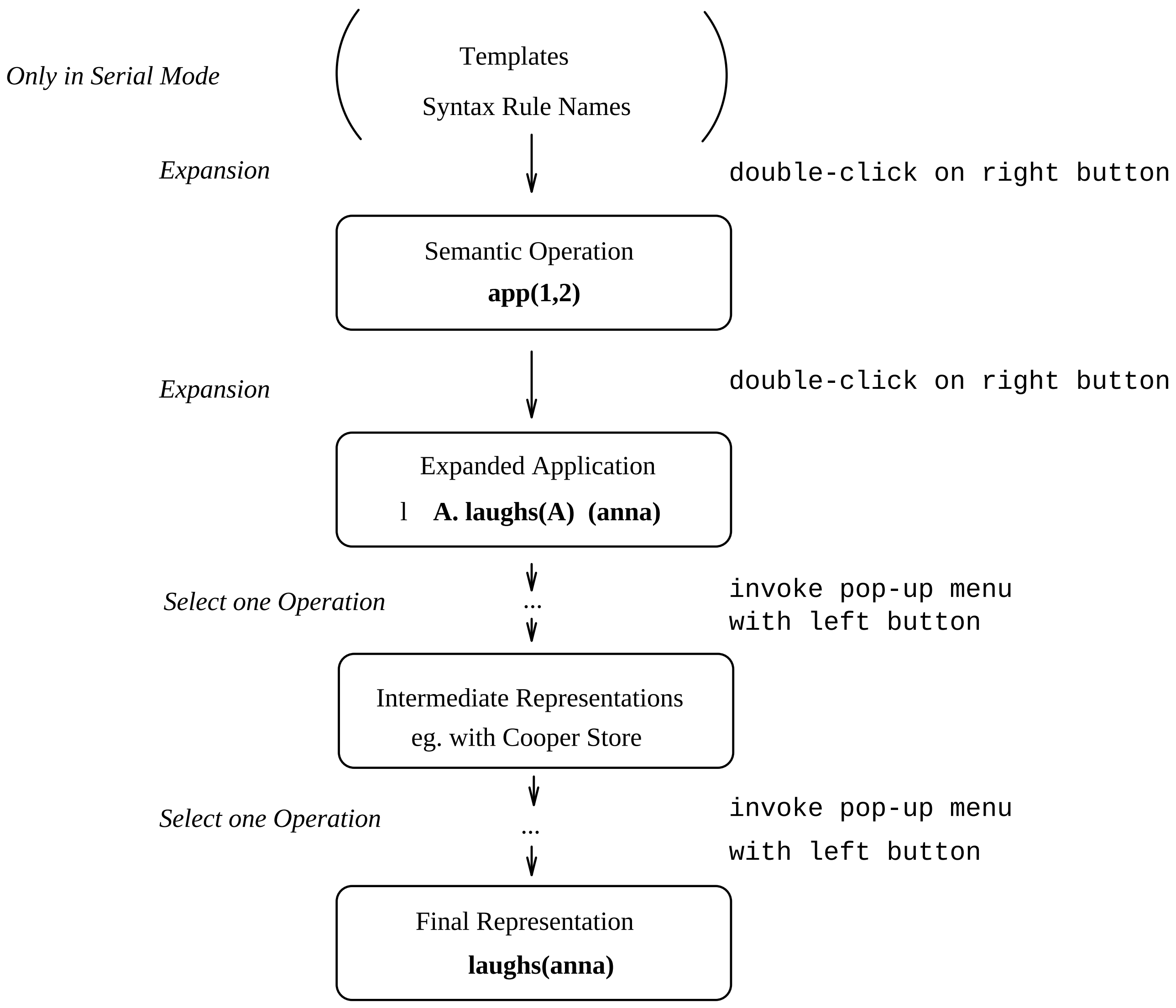


Figure 2.16: How to interactively obtain a semantic representation

The Parameter Menu

In the parameter menu you will find all linguistically relevant parameters. Each of these is connected to a cascaded menu from which you have to choose one particular setting. Note that not all possible combinations of parameter settings are also feasible. In Section 2.1.4 we present incompatible settings.

Semantics lets you choose the semantic representation formalisms that are currently available in *CLEARS* .

- **DRT** Implementation of the first two chapters of [Kamp and Reyle, 1993]
- **Lambda-DRT** λ-DRT see [Bos *et al.*, 1994; Kuschert, 1995]
- **Compositional DRT** A variant of a compositional DRT by [Muskens, 1994]
- **Gen. Quantifier** Language of Generalised Quantifiers
- **Intensional Logic** intensional Montagovian-style fragment based on [Montague, 1973]
- **Lambda Language** extensional Montagovian-style fragment based on [Dowty *et al.*, 1981]
- **Situation Semantics** based on the fragment described in D15/D16

Grammar lets you choose a grammar.

- **Grammar 1** is a very simple DCG grammar
- **Grammar 2** is a DCG style grammar with features
- **DRT Grammar** is a GPSG-style Grammar according to chapter 0 of [Kamp and Reyle, 1993].

Lexicon lets you select a lexicon for your grammar. Lexicon and Grammar have to agree, so you should always take care to change Lexicon and Grammar together.

- **Lexicon 1** is suitable for Grammar 1
- **Lexicon 2** is suitable for Grammar 2
- **DRT Lexicon** Coverage of the first two chapters of [Kamp and Reyle, 1993]. Only suitable for DRT-Grammar.

Parser lets you select a Parser for your Grammar. In the present system the choice of the grammar determines the parser.

- **DCG** for Grammar 1 and 2
- **Chart** Parser for DRT fragment

Typing lets you choose between typed and untyped λ-calculus.

Untyped for untyped λ-Calculus

Typed for typed λ-Calculus

SynSem Mapping describes the mapping between the grammar rules and how they are linked to semantic operations. This only shows effect when used in *serial* or *radical-serial* mode.

Rule2Rule enables Rule to Rule syntax semantics mapping. Grammar rules are linked to semantic operations.

Templates enables template based syntax to semantics mapping.

SynSem Mode describes the way in which the syntax tree is annotated with information used to extract semantic information.

Serial annotates the syntax tree with templates or syntactic rules that are described in 2.1.4. The semantic operations connected with these are then fetched in a subsequent step.

Parallel annotates the syntax tree immediately with semantic operations.

Radical Serial doesn't annotate the tree. Used together with typed λ-calculus for type-driven semantic construction.

SUBJ-VP Application enables the application order of a VP and its subject at an S node. This allows to treat the meaning of an NP either as an entity or or as a higher typed object.

Apply(Subj, VP) applies the VP to the complement daughter at an S node.

Apply(VP, Subj) applies the complement daughter to the VP at an S node.

Logical Variables lets you choose the kind of variables for the λ-reducer to work with.

Logical Variables enables logical variables implemented as Prolog variables.

Constants enables logical variables implemented as constant terms, realised via the Prolog predicate `numbervars/3`.

Lambda Reducer lets you choose the computing strategy of our λ-reducers.

Unification Based enables unification based beta reduction.

Modified Unification enables β reduction using a copying stage ([Cooper *et al.*, 1993])

Substitution Based enables full rewrite based lambda reduction including alpha conversion

Quantifier Store lets you choose the strategy of quantifier storing. The names of the parameters should be self explanatory.

Cooper Storage see [Cooper, 1983]

Nested Cooper Storage see [Keller, 1988]

None

Get Defaults selects a default parameter setting for a semantic representation. The user can modify the default settings in the file `$FRACAS/user_defaults.pl`, but this should only be done by the experienced user.

Check Setting checks your actual parameter setting.

Dependencies between Parameters

Not all possible combinations of our parameters lead to any output at all. These incompatibilities fall into two classes, logical incompatibility or implementational incompatibility. The latter class describes possibilities that are not yet implemented, as opposed to the former class, in which there are logical reasons for their incompatibility. Whereas parameter combinations that fall into the latter class may be allowed in future releases, the ones that fall into the former class will (very likely) never lead to any workable configuration of the system. The implementational problem at this stage concerns the *Situation Semantics* only working with *Logical Variables*.

For the class of logical incompatibilities we present tabulars with positive occurrences of parameter combinations. The negative cases immediately follow from the fact, that these positive cases describe exhaustively all possible combinations of one pairing. But this also implies that all other settings are freely combinable.

Semantics	**SUBJ-VP Application**
Lambda-DRT, Compositional DRT, Intensional Logic Lambda Language, Situation Semantics	apply(subj_vp)
Gen. Quantifier	apply(vp_subj)

Semantics	**SynSem Mapping**
Situation Semantics	Rule2Rule

Grammar	**Lexicon**
Grammar 1	Lexicon 1
Grammar 2	Lexicon 2
DRT Grammar	DRT Lexicon

Lambda Reducer	**Logical Variables**
Unification Based Modified Unification	Logical Variables

One negative instance has to be mentioned: The *untyped* λ-calculus only works with the SynSem Mode *rad_ser*.

The Graphics Menu

The Graphics menu involves parameters that are concerned with the graphical output or with manipulating the construction process. Almost all entries are checkbuttons that allow to choose between two possibilities. The last entry allows to supply the changed parameters to the graphics on the fly.

Draw Stack If set, draws all available semantic information of one node stacked on top of each other.

Draw Boxes If set, draws boxes around semantic information.

Storage Sequence If set, draws the Quantifier store as a stack. If not set draws it as a sequence.

Full Representation If set, calculates the semantic representation of the whole tree in one step.

Immediate Info If set, a mother takes only the available semantic information of the daughters to combine the meaning. If not it calculates the semantic representation for each daughter before calculating her own representation.

Skip Sem. Operation Only useful in Serial Mode. If set, allows to go from a Template/Rule annotation of a node immediately to the semantic representation without showing the semantic operation.

Change Graphics Change Graphic parameters for the output on the fly.

The Tools Menu

Here you find useful tools to compare different semantic formalisms and different output.

Translate Allows to translate between different semantic formalisms. This is not implemented for all possibilities.

Add Translation Temporarily adds a translation in stacked and boxed form to the actual semantic output.

Copy Structure Starts a new window with a copy of the actual output.

New CLEARS Starts a fresh *CLEARS* process

Add-ons in the Help menu

About CLEARS Information about *CLEARS* .

Authors Information about the authors.

2.1.5 Pop-Up Menus

With several structures of the output representation one can associate Pop-Up Menus which get activated while clicking on that structure. One recognises a clickable structure on the change of the mouse cursor from an arrow to a hand, while moving over these regions. Pop-up menus are invoked by a single mouse-click, operations that lie usually on the right mouse button are started with a double-click. On the left mouse button one usually finds operations connected with further linguistic processing such as Application or Discharge. The middle button is used for manipulating the structure and the tree graphically, such as zooming or editing. The pop-up menus may also vary with the actual semantic formalism or the actual grammar, so we describe a superset of all possible menu entries, although there may be no such menu that involves all these commands.

The clickable structures include the *semantic nodes*, the *syntactic nodes*, the *lexical nodes* (syntax and semantic information) and the *quantifier store*.

Pop-Up Menu for Semantic Nodes

Linguistic Processing holds the commands to drive the composition process, allowing to choose an individual operation at each point in the derivation. .

- **Construct Meaning** allows different steps of the composition process to be done at once. This command also lies on the right mouse button for convenience.
- **Merge** merges together two DRSs.
- **Reduce** just reduces an Application.
- **DownUp Cancel** invokes the DownUp Cancellation.
- **Meaning Postulate** allows to select and apply a meaning postulate.
- **Store** will store a representation in the store.

Editing offers commands to edit representations or to manipulate the look of the graphical output.

- **Edit** allows to edit a representation
- **Undo** allows to undo your last operation.

Zoom in displays only the selected semantic node on the screen.

Zoom out goes back to the display the whole tree.

Pop-Up Menu for Syntax Nodes

Again we have one menu dealing with tools for further inspection of the syntax node on the left mouse button, whereas the*Editing* menu lies on the middle button. The right button is not used in this version.

Inspection holds some commands for examining the syntactic features, provided that the grammar exploits such features.

- **Expand FS** allows that the feature structure of that node is shown instead of the plain category
- **Expand Selection** only selected features are displayed.
- **Select Feature** allows to select a subset of all features. This may be used to display only those features relevant to the semantic phenomena under examination.
- **Show Category** displays the category of a node instead of the feature structure.

Editing contains the same entries as the one for semantic nodes. Therefore we wont give an explanation here, but refer to 2.1.5.

Pop-Up Menu for Lexical Nodes

These nodes only have a subset of Operations that one finds at the corresponding non-terminal nodes.

Pop-Up Menu for Quantifier Store

Fore these objects, there is only one menus lying on the left mouse button.

Discharge takes the variable out of the store and substitutes its occurrence in the main-formula with its associated, stored formula.

Edit allows to edit the content of the store.

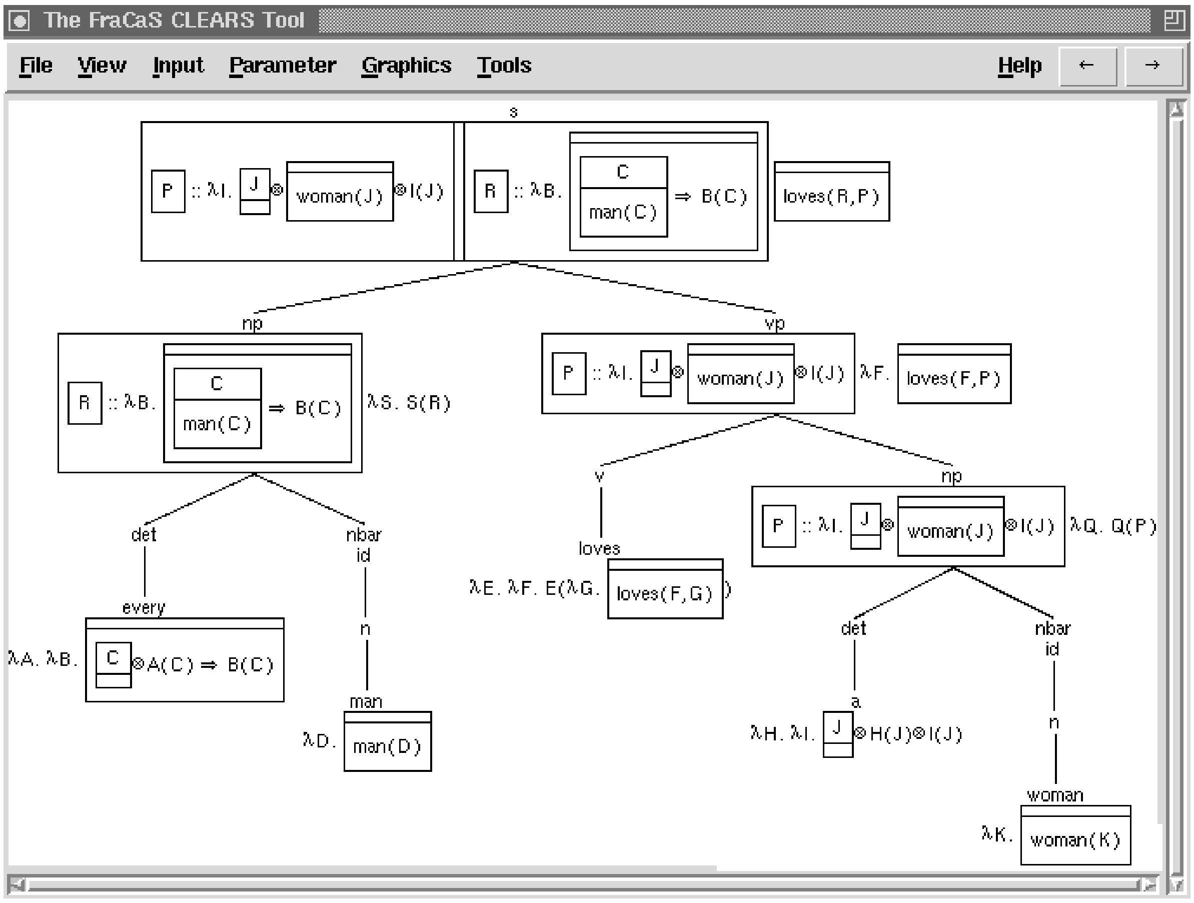


Figure 2.17: Fully expanded representation for `every man loves a woman`

2.1.6 Selected Topics

Here we show how use the possibilities that *CLEARS* offers for certain representative tasks in a teaching or researching environment.

Dealing with Disambiguity

We go back to our example in Section 2.1.3 and show how you can obtain both readings of that sentence and displaying them at the same time on your screen. Assume you have constructed a semantic representation of the sentence `every man loves a woman`, up to the `s` node, so that your present output looks like in Figure 2.17. With the command **Copy Structure** from the **Tools** menu, we can create a copy of our actual output and display this copy in a new *CLEARS* window. Figure 2.18 shows both *CLEARS* processes with a copy

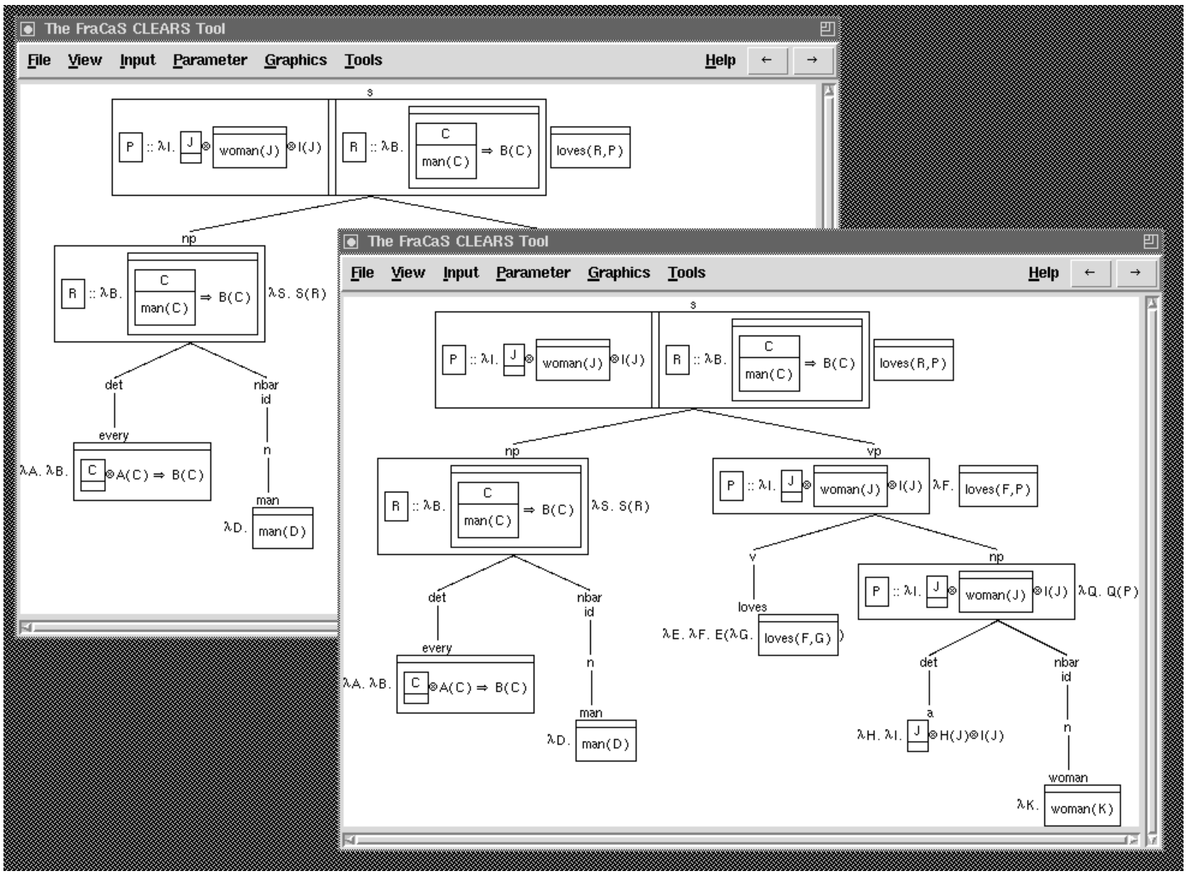


Figure 2.18: Two *CLEARS* processes displaying the same representation

of the same graphical output. So we have the possibility to choose for each window a different order to discharge the variables in the store. This allows us to display both readings at the same time. The final output is shown in 2.19. To display only the relevant semantic nodes in both processes, we choose the **Zoom_in** command from the editing[14] pop-up menu for semantic nodes.

Translation

Now we can also translate from one representation formalism to the other. In the current version, we have only implemented translation routines from the DR Theories to the ones based on predicate calculus. We follow the algorithm outlined in [Kamp and Reyle, 1993]. For our final result of the last section, shown in Figure 2.19 we show a translation to our **Lambda Language** in Figure 2.20.

[14]The one you get clicking on the middle button, if the mouse is over a semantic node.

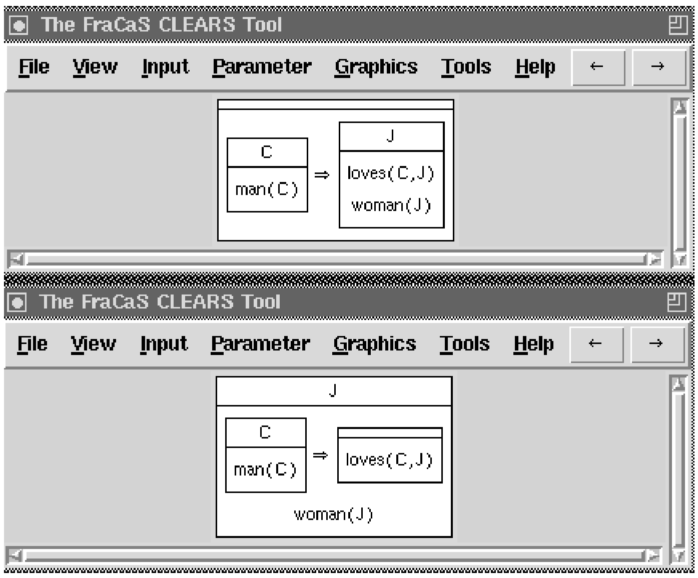

The FraCaS CLEARS Tool
File View Input Parameter Graphics Tools Help ← →
C
man(C)
⇒
J
loves(C,J)
woman(J)
The FraCaS CLEARS Tool
File View Input Parameter Graphics Tools Help ← →
J
C
man(C)
⇒
loves(C,J)
woman(J)

Figure 2.19: The two final representations

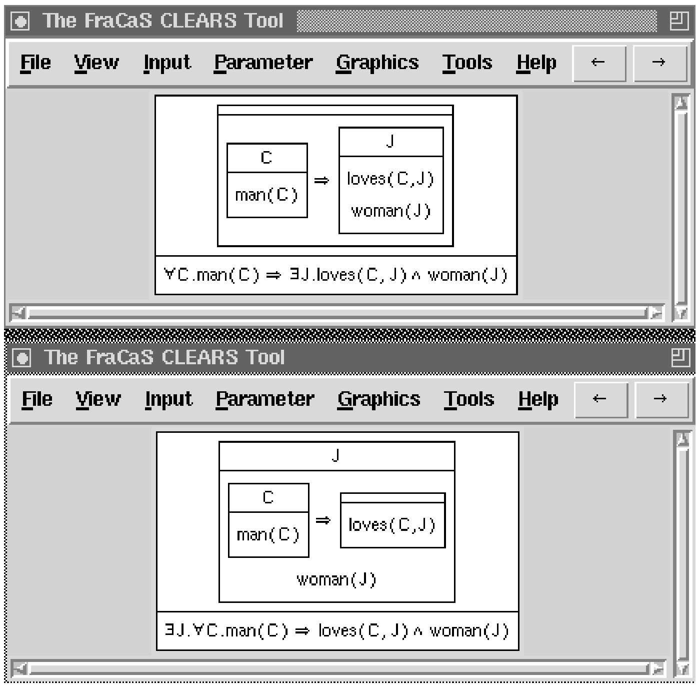

The FraCaS CLEARS Tool
File View Input Parameter Graphics Tools Help ← →
C
man(C)
⇒
J
loves(C,J)
woman(J)
∀C.man(C) ⇒ ∃J.loves(C, J) ∧ woman(J)
The FraCaS CLEARS Tool
File View Input Parameter Graphics Tools Help ← →
J
C
man(C)
⇒
loves(C,J)
woman(J)
∃J.∀C.man(C) ⇒ loves(C, J) ∧ woman(J)

Figure 2.20: Representations and Translations

2.1.7 Appendices

Starting Clig

Here we present the different commands at the SICStus Prolog top-level. Always make sure that you have set the appropriate Environment variables. Check 2.1.7 for this.

clears
: starts the *CLEARS* system with default values for λ-DRT. You might change this default setting in the file `$FRACAS/Clears/default_values.pl`

clears(+Sems)
: starts the *CLEARS* system with default values for the semantic representation Sems. An overview of possible values is given in 2.1.7. Choose a value under the *sems* value in that table.

h
: shows a help screen, that shows you the possible commands.

Environment Variables

We assume that $FRACAS is the name of the directory in which you have installed *CLEARS* . $LIBRARY is the directory in which you find the tcl/tk libraries on your local machine.

```
setenv CLIG_DIRS $FRACAS/Clig
setenv CLEARS_DIRS $FRACAS
setenv TCL_LIBRARY $LIBRARY/tcl7.4
setenv TCL_LIBRARY $LIBRARY/tk4.0
setenv LD_LIBRARY_PATH ...$LIBRARY/tcl7.4:$LIBRARY/tk4.0
```

Library Parameters

Here we present an tabular showing which library parameters are set when selecting from the *CLEARS* menu **Parameters**.

	Parameter Menu	Library Parameter
Semantic Formalism	Semantics	sems
	DRT Lambda-DRT Compositional-DRT Gen. Quantifier Intensional Logic Lambda Language	drt ldrt cdrt lgq il ll
Different Grammars	Grammar	grammar
	Grammar 1 Grammar 2 DRT Grammar	psg1 psg2 drt_gram
Different Lexica	Lexicon	lexicon
	Lexicon 1 Lexicon 2 DRT Lexicon	lex1 lex2 drt_lex
Different Parsers	Parser	parser
	DCG Chart	dcg chart
typed vs. untyped calculus	Typing	typing
	Typed Untyped	t ut
Syntax-Semantic Mapping	SynSem Mapping	synsem
	Rule2Rule Templates	rr tm
Annotation of Syntax Trees	SynSem Mode	serpar
	Serial Parallel Radical Serial	ser ppar rad_ser
Dealing with type-raised NPs	Subj-VP Application	subjvp

	Apply(Subj, VP)
	Apply(VP, Subj)
		s_vp
		vp_s

different style of variables	Logical Variables	logvars
	Logical Variables Constants	pv nc
Computation of β-reduction	Lambda Reducer	lambdared
	Unification Based Modified Unification Substitution Based	un clb sub
selects the quantifier storage	Quantifier Scope	qscope
	Cooper Storage Nested Cooper Storage None	cs ncs no

2.2 Extending CLIG with Interaction and User Defined Graphics

D15 describes the CLIG grapher and its use as a general tool for visualizing common linguistic graphical notations.[15] The main purpose of the program is its use as a graphical interface for applications in computational linguistics. These applications can display their data by creating *description strings* and sending them to the grapher. These strings are hierarchical textual descriptions of the graphical structures. E.g., the following text is the description for the graphical structure seen in Figure 2.21:

```
{drs {plain-text B}
     {tree {plain-text s}
           {plain-text B}
           {tree {plain-text vp1}
                 {tree {plain-text vp}
                       {tree {plain-text v}
                             {plain-text owns}}
                       {tree {plain-text np}
                              {tree {plain-text pn}
                                    {plain-text ulysses}}}}}}
      {plain-text jones(B)}}
```

With its description string language, CLIG provides a wide range of possibilities to create the graphical representations one has in mind. Objects can e.g. be ordered in stacks or sequences, can be put in boxes or can each be assigned colors separately. The basic notations like trees, texts or DRS boxes can be nested freely within each other, producing complex graphical structures like e.g. Figure 2.22. However, for certain applications, the grapher's built-in facilities might not be sufficient to produce the desired output. However, CLIG has been designed in a way that makes it relatively easy to add new graphical features. Section 2.2.2 describes how a user can add his or her own graphical objects to the grapher.

By linking objects in the description string to pieces of code, an application defines *interactive* graphics where a user can perform application-dependent actions by clicking on objects with the mouse cursor. Interactive graphics are under the control of the application which creates the description strings. The application decides which objects behave in what way by sending the appropriate code with their descriptions. Interaction between grapher and application is freely programmable and therefore very flexible, but may also require some effort on the side of the programmer of the application. CLIG is extendible

[15]This section is a preliminary version of a user manual to appear in [Konrad, 1996].

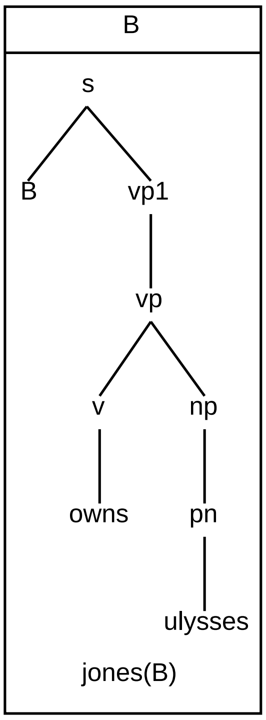


Figure 2.21: A DRS with a tree nested inside.

in the sense that the application programmer can add the amount of interactive behavior needed for the specific application. Section 2.2.1 explains how interfaces between applications and the grapher can be build.

Tcl/Tk as the programming environment

Clig has been implemented using the software package Tcl/Tk, a programming system for graphical user interfaces. Tcl, the *tool command language*, is a simple interpreted scripting language providing basic programming facilities like variables, procedure definitions and loops. Tk is a toolkit for the X Windows System that extends the core Tcl-language with commands to build Motif-like graphical user interfaces. The Tcl/Tk package can be used via the interpreter program Wish that contains the language interpreter for Tcl and the Tk toolkit. Clig's source code is a set of Tcl scripts which gets interpreted by Wish.

The use of Tcl/Tk has some obvious advantages and one minor drawback. One advantage is portability: Tcl/Tk is free and widely available for different OS-platforms. The X-Windows functionality used by TK gets emulated for foreign systems like Windows or Mac OS. Another advantage is the *rapid prototyping* of code. Since TCL gets interpreted, code extensions or modifications for the grapher can be loaded on-the-fly without the need for recompilations. Modified code simply overwrites old, while new code can be added following CLIG's simple code conventions (see section 2.2.2). In contrast to C or C++, TCL/TK hides low-level details of the X-Windows environment from its users, making it generally easier to write TCL/TK graphical user interfaces than C/C++ user interfaces. The minor disadvantage of TCL/TK is the slower execution speed of the interpreted TCL code which can be circumvented by writing C routines for time-critical algorithms. Thanks to the speed of modern work stations, this kind of optimization is rarely necessary. CLIG currently does not contain any non-TCL code.

2.2.1 Interaction

An interactive graphical structure contains parts which are mouse sensitive; in CLIG these parts are linked to small TCL programs called *scripts*. A TCL script in an interactive graphical structure could e.g. open a pop-up menu or ask an application for more information on whatever a user has clicked on. The linking of code to graphical objects in the description strings can be done by the special commands `clickable` and `active`.

The `clickable` command

The `clickable` command uses one single script which gets executed when the user *double-clicks* on the associated object. The command takes two arguments, the object to be drawn and the TCL script linked to it. The following example creates a mouse sensitive text box which produces a short message when clicked by the user:

```
{clickable {bigbox plain-text "click me!"}
           {puts "YOU CLICKED ME!"}}
```

The `active` command

The `active` command can define multiple possibilities for the same object or define special behaviors. It allows the use of several event-script pairs for the same object. An *event* can be any X-Window event known in TK such as `<Leave>`, `<1>`, `<Double-3>` etc.

The general syntax for `active` is

```
{active <object> {<event> <script>} ... {<event> <script>}}
```

If you wanted the message from the last example to appear when the middle mouse button gets clicked once, you would express it like this:

```
{active {bigbox plain-text "click me!"}
        {<2> {puts "YOU CLICKED ME!"}}}
```

Application meets Clig

The concept of interfaces between applications and Clig is shown in Figure 2.23. The application "talks" to the grapher by sending it description strings. Some systems like Sicstus3 have their own Tcl/Tk interface which can be used for this. Another way to create an interface to the grapher is the use of the **send** mechanism which can be used by any Tk application. We are also currently working on a very simple interface which uses Clig's standard input and output. In all these cases, the main drawing routine `clig` gets called by the application with an description string containing `clickable` or `active` commands. The commands use Tcl scripts which in response to a mouse action force the application to do something. The following example is taken from the larger description string which produces the graphical structure in Figure 2.22:

```
{bigbox clickable
{plain-text "app(1,2)"}
{prolog_event "click(sem_node, d1, 3)"}}
```

The example structure is created by the educational tool developed in the FraCaS project. The tool uses the Tcl/Tk interface provided by Sicstus3 Prolog. In the example, the text `app(1,2)` is linked to a script specifying that a `prolog_event` command gets executed when the text gets double-clicked with the mouse. The `prolog_event` command results in an event which gets noticed by a loop in Prolog waiting for such an event. The string `click(sem_node, d1, 3)` informs Prolog that the node `3` has been clicked on. The main loop then can take the appropriate action and will display a new graph where node 1 has been applied to node 2.

The concrete realization of the interface between an application and Clig depends on what the host system offers as a means of communicating with Tcl/Tk. We are currently experimenting with different general approaches like using a TCP/IP channel or simple standard input/output piping to allow a common interface.

2.2.2 Adding graphical objects

This section describes how user defined objects can be included into the grapher. Subsection 2.2.2 explains how code extensions appear in the description strings which are used for all representation and interfacing purposes within the grapher. Subsection 2.2.2 explains the conventions necessary for code extensions.

Description strings

The description of a graphical object in CLIG is done with a hierarchical list in TCL-syntax, the *description string*. The general syntax of a description string is

```
{<command> <parameter_1> ... <parameter_n>}
```

Most commands take description strings as their parameters. If a command takes only one description as parameter, the command is said to *modify* the description. The general syntax for such a modifying command is

```
{<modify-command> <command> <parameter_1> ... <parameter_n>}
```

E.g. a description string like `{neg plain-text "x"}` is a Tcl-list with 3 members, with the first two of them being commands. The `neg` command modifies the output of the `plain-text` command by putting a negation in front of the text.

Whenever CLIG draws a graph, it really interprets a description string like this as a piece of TCL-code which lacks some information, e.g. the exact positioning of the objects. The description string gets executed by an internal interpreter which recursively fills in the missing information after it has calculated the correct positions of all objects in the drawing area. Each command in the description string has an equivalent TCL procedure which gets called with the missing information and all parameters. Extending CLIG means writing such procedures and adding them to the source code.

As an example, take a look at the code which is responsible for drawing negations:

```
proc neg {x y where mtags obj} { ;# draw negated drs
        seq $x $y $where $mtags \
        [list {sign neg} {plain-text " "} $obj]}
```

Without going into details, the code for `neg` simply calls another command used in CLIG, `seq`, with three graphical objects, the negation sign, a space character and the object which should be negated. The `seq` command gets some additional parameters it inherits from the `neg`-call, namely x and y for the exact position of the object, *where* for the canvas which should be used (CLIG uses two canvases, one for actual drawing and one for calculating the size of an object) and *mtags* which is used for active (clickable) regions. The *mtags* parameter contains a set of labels and usually is only manipulated by the `active` and `clickable` procedures for grouping together objects with the same scripts linked to them. All other commands do not change *mtags* but must inherit this parameter to each of its daughters. By inheriting labels to their daughters, clickable objects correctly execute their associated script when one of their daughter objects are clicked on.

Code conventions

Each graphical command for CLIG must have exactly 5 parameters: the x and y positions of the top left corner, the *where* canvas, the *mtags* parameter and an additional parameter *obj* containing everything that was in the object description string except the command name itself.

Each command must return the width and height of the drawn object as a list. A good example of the use of the sizes would be the code for `underline` which simply underlines another object:

```
proc underline {x y where mtag obj} {
        ;# drawing
        set size [execobj $obj $x $y $where $mtag]
        ;#
        set yline [expr $y+[lindex $size 1]+2]
        $where create line $x $yline \
                           [expr $x+[lindex $size 0]] $yline
        ;# returning the complete size of the underlined
        ;# object:
        list [lindex $size 0] [expr $yline-$y]}
```

The `underline` procedure first draws the object *obj* by calling `execobj`, the main drawing function which tries to execute the description string *obj* at position x,y. `execobj`, the interpreter procedure, returns the width and height of the object which `underline` uses to calculate where (y position + 2 points) and with which width to draw the line under the object. `underline` returns the width of the object and the height + 2 points as the size of the object drawn.

This example covers almost everything one has to know about writing code for extending the grapher. A procedure has to have a unique name, must use the 5 formal parameters described above and must return the correct width and height of the object it draws—and then it is a legal extension of the grapher. An application can simply load the additional code using the methods described in section 2.2.2 and then use the new procedures in the same way as the "factory" commands CLIG provides. A user can even replace the built-in commands by simply overwriting them with his own code since TCL will not report an error for redefining procedures. In this way, an application might modify the graphical output for its special needs, e.g. by either optimizing for beautiful layout or speed, without the need for actually changing the original code.

The simple convention used here has a few practical drawbacks, e.g., the layout algorithm of an object can only use rectangular regions for calculating the positions of its daughters. In the case of objects with very irregular shape, this may result in a waste of space and/or may result in an aesthetically unsatisfying result. Another drawback ist the strict top-down approach of drawing the graphics. Some graphical structures must calculate the size of their daughters before they can actually draw them; in this case a bottom-up approach would be better. However, the nasty side effect of having to draw objects unnecessarily because of repeated size calculations with CLIG's top-down algorithm can be circumvented by using the `calcsize` command described in the next section.

Canvases and calculating sizes

Sometimes a command has to know the size of an object *before* it gets drawn. E.g., the `stack` command stacks several objects on top of each other; the command uses the size of the biggest box as the maximum x size of the whole object and centers all objects within this maximum size. Therefore, before any object gets drawn, `stack` calculates all sizes and stores the maximum in the variable $xsize$. A simplified version of this calculation routine would look like this:

```
foreach item $stack {
        set size [calcsize $item]
        set xsize [max $xsize [lindex $size 0]]}
```

The procedure which calculates the size of all objects, `calcsize`, actually draws the object on an invisible canvas and returns the size which every CLIG drawing procedure returns.

The `calcsize` procedure is almost identical to `execobj` except that it does not need any coordinates nor *mtags* which only play a role in the visible drawing area. While `execobj` uses the visible *.graphics* canvas, `calcsize` simply uses the canvas *.invisible*. This special canvas never gets displayed by the grapher; it only exists for evaluating sizes. Procedures can take advantage of this distinction by minimizing what actually gets drawn during size evaluation. Since drawing in an invisible area does not have to look good or even complete, procedures may leave out any actual drawing whenever they encounter `.invisible` as their *where* parameter, as long as they can calculate the correct size of the object they draw.

As an additional optimization, `calcsize` uses a *memoization* technique for storing sizes it already has calculated.

Including your own code

To include your own code into CLIG, you can simply change the file extend.tcl in the CLIG-directory so that it loads your files when you start the grapher. The only problem with this is that the extensions get loaded whether they are needed or not and that you have to modify `extend.tcl` whenever you install the grapher. An application using CLIG usually can do better by simply performing the `source` command of TCL. In this way, an extension gets loaded only when needed by the specific application.

An example: Circles

Supposed that you wanted to write a package for CLIG for displaying e.g. *finite state machines* and therefore needed circles to represent the states of the machine. In this case you would have to write your own code, since circles are not supported in the standard release of CLIG.

The first step would be to create a new file `circle.tcl` and to add this file to the `extend.tcl` loading list as described in section 2.2.2. The file `circles.tcl` will be loaded whenever you run CLIG. Every time you change your code, you can include your changes simply by exiting your old CLIG and restarting it again. An even easier way would be to use the **Load...** option from the CLIG main menu. Although this option usually is used to load object descriptions, it also is able to load arbitrary TCL source code.

Now you can add the circle code. You want to use the commands `state` for a normal state and `final` for a final state in your finite state package. A state simply consists of a circle of a certain size, while a final state will be displayed as two concentric circles. Additionally, you want to have states to have arbitrary sizes, so that `{final 20}` will display a finite state with a radius of 20 points.

The code for a normal state could look like this:

```
    procedure state {x y where mtags rad} {
        set rad_size [expr $rad*2]
        $where create oval $x $y \
                           [expr $x+$rad_size] \
                           [expr $y+$rad_size] \
                           -outline black -tags $mtags
        list $rad_size $rad_size}
```

Since you wanted the state to have a radius of *rad*, the size of the complete object will be 2 *rad* in x and 2 *rad* in y direction. The procedure draws an oval with a black outline which bears the labels *mtags*. The labels are necessary for interactive graphs, see section 2.2.1. The code does draw an oval whenever the procedure gets called, but this may not always be necessary. Trees and other complex structures often check for the size of their daughters just to see where they can be placed. The following refinement of the code leaves out this unnecessary drawing of the circle:

```
proc state {x y where mtags rad} {
        set rad_size [expr $rad*2]
        if {$where!=".invisible"} {
           $where create oval $x $y \
                              [expr $x+$rad_size] \
                              [expr $y+$rad_size] \
                              -outline black -tags $mtags}
        list $rad_size $rad_size}
```

This version only draws the circle when the destination canvas is not equal to `.invisible`. The code for final states is very similar to the code for states; it just puts a smaller circle inside the outer circle.

```
proc final {x y where mtags rad} {
        set rad_size [expr $rad*2]
        set offset 3
        if {$where!=".invisible"} {
           $where create oval $x $y \
                              [expr $x+$rad_size] \
                              [expr $y+$rad_size] \
                              -outline black -tags $mtags
           $where create oval [expr $x+$offset] \
                              [expr $y+$offset] \
                              [expr $x+$rad_size-$offset] \
                              [expr $y+$rad_size-$offset] \
                              -outline black -tags $mtags}
        list $rad_size $rad_size}
```

After rerunning CLIG, you should have the possibility to display states and final states as displayed. For a quick test, just type in the following test file in a text editor and load it with CLIG:

```
# example: two states
clig {Seq {state 20}
          {final 20}}
```

Colors

The `color` command provides an easy way to add color information to graphs. User defined object code must include the following to be compatible with the `color` command:

- the command `globals main_color` to make the global variable `main_color` known to the procedure
- each drawing command (like e.g. drawing a line) must use `main_color` for the main drawing color. Most drawing commands, e.g. `create line` use `-fill` as the option for specifying the foreground color.

The code for `underline` contains only one drawing command, the `create line` code which draws the underlining line. With `-fill $main_color`, the line's color can be changed by a `color` command:

```
proc underline {x y where mtag obj} { ;# line under object
        global main_color
        set size [execobj $obj $x $y $where $mtag]
        set yline [expr $y+[lindex $size 1]+1]
        $where create line $x $yline \
                           [expr $x+[lindex $size 0]] $yline \
                           -tags $mtag -fill $main_color
        list [lindex $size 0] [expr $yline-$y]}
```

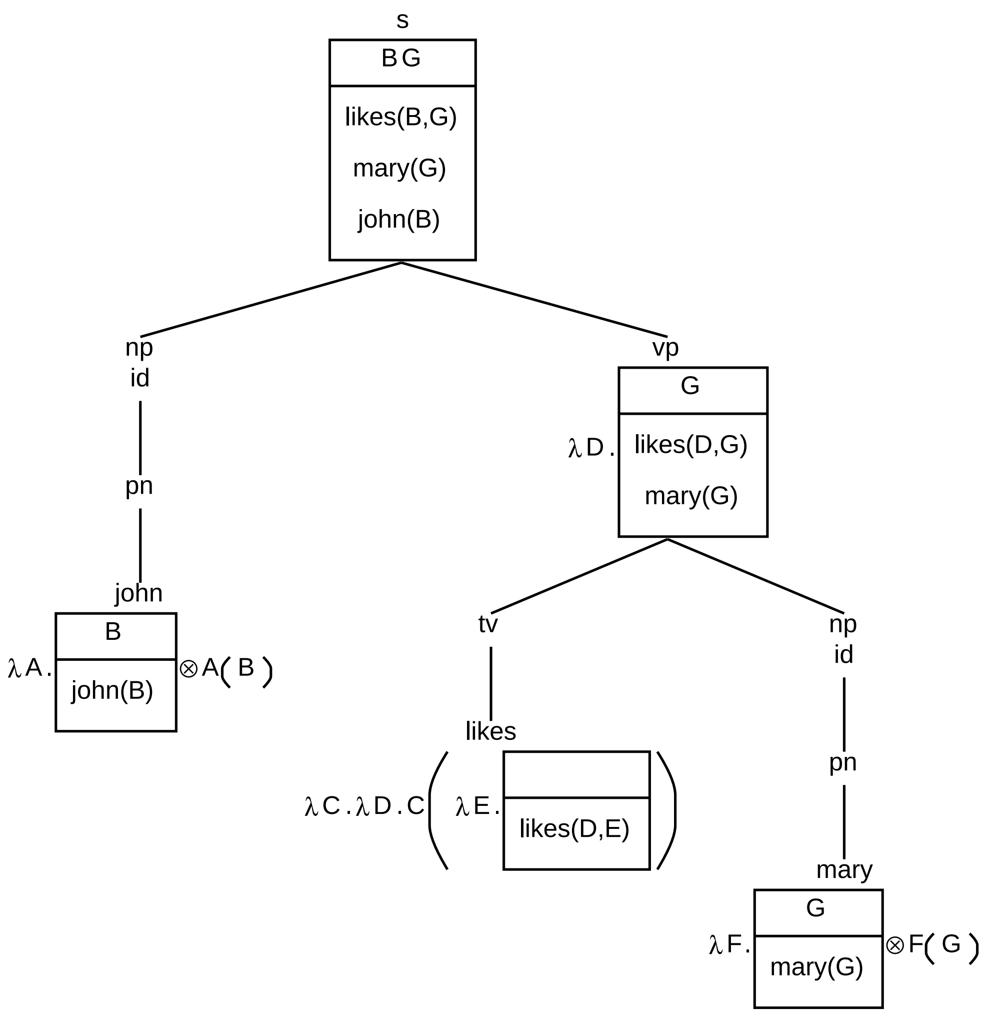


Figure 2.22: A complex nested graphical structure

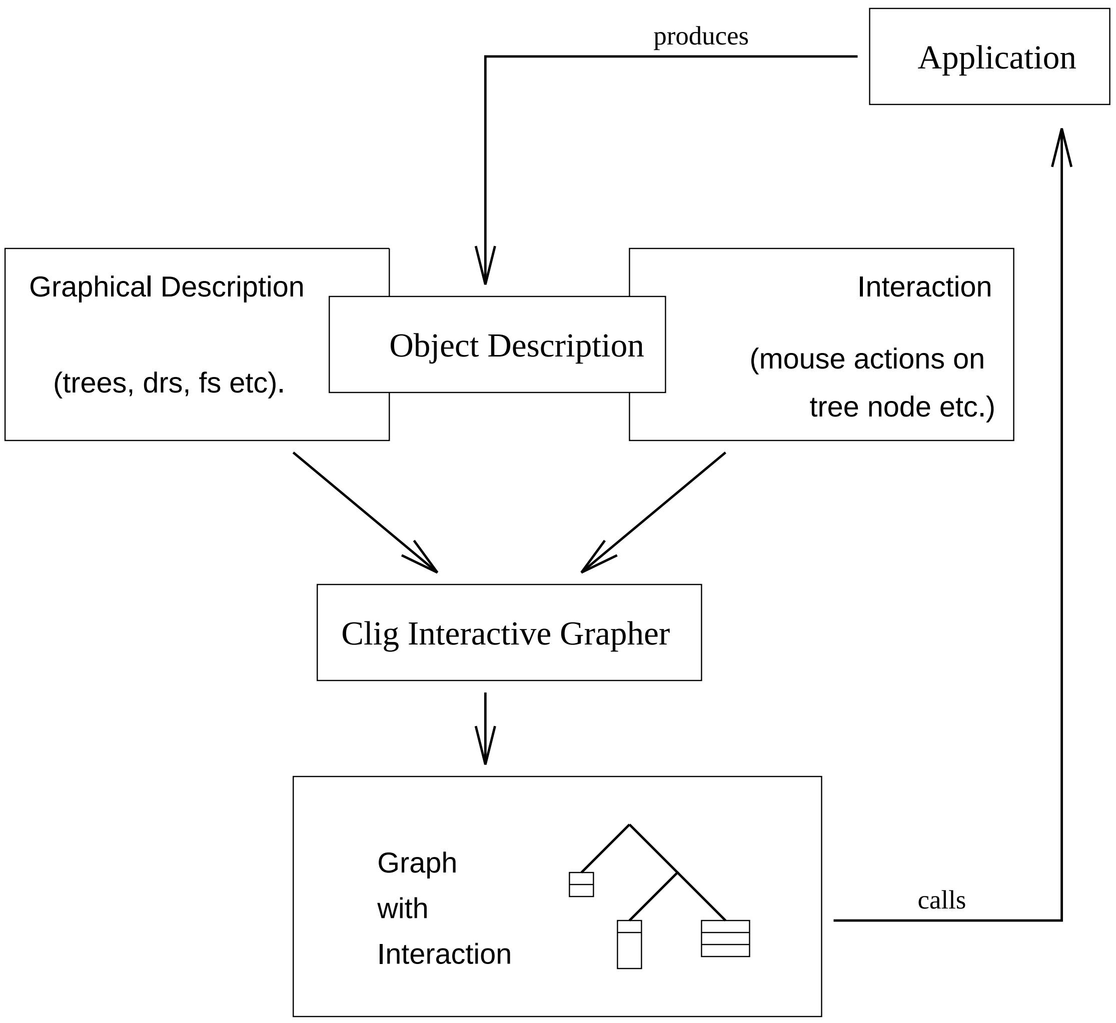


Figure 2.23: Application-to-Clig interaction.

2.3 Ekntool User Manual

2.3.1 Introduction

EKNTOOL is a tool supporting the development of grammars producing semantic output in EKN notation. The tool includes programs to perform operations on EKN-style representations and to convert these representations in Latex format and a `perl` interface that invokes Prolog to parse a sentence producing an output in EKN notation, puts the output in Latex format, and displays the result using xdvi. The tool comes with several demonstration grammars, including the 'fracas main' grammar.

EKNTOOL derives from the STDRT tool developed by Robin Cooper, Julian Day and Phil Kime for DYANA-2. Unlike STDRT, ekntool allows different grammars to be used, and new grammars are included. In particular, a new default grammar is specified - the situation-theoretic grammar developed by Robin Cooper and Massimo Poesio as part of FRACAS project. Many changes have also been made to the original code to make the system more usable by the general public and more portable to other sites.

The tool has been tested under SunOS 4, using `sicstus` version 2.1, `latex` 2e, and `perl` version 5.

2.3.2 Interacting with ekntool

Starting Ekntool

You can start the `perl` interface by calling the `perl` script `ekntool` contained in the directory which contains the code. After establishing a connection with prolog, the program prompts the user for some input, which can be either a sentence or a command (see below).

You can also use the system directly from Prolog. To do so, start `sicstus`, load the file `main.pl` contained in the ekntool home directory, and then type

```
?- loop.
```

This will start a loop during which of reading sentences and displaying the output. The output is displayed in the same form used by the `perl` script.

The Perl Interpreter

After typing EKNTOOL, the user is presented with a prompt. The `perl` interpreter can be used both to parse sentences and to execute commands (to set switches, get help, and so forth).

In order to have the system process a sentence, simply type the sentence at the prompt. Single sentences can terminate either with punctuation marks (currently, only period, '.', is recognized) or with carriage return. The sentences in a multiple sentence texts should be separated by punctuation marks.

Typing 'help' will show you what commands are available. Currently (January 1996) these include:

`'help'`	`Displays this summary`
`'test'`	`Tests entire current database. See help on 'test'`
`'help <subject>'`	`Displays help on 'subject'`
`'load <file>'`	`Loads database of discourses 'file'`
`'loadpln <file>'`	`Loads PLN database of PLN objects 'file'`
`'view'`	`View currently loaded database`
`'viewpln'`	`View currently loaded PLN database`
`'set <switch> <value>'`	`Set 'switch' to 'value'`
`'set'`	`Show current switch settings`
`'reload' or 'rl'`	`Reload prolog sources and lexicon`
`'print'`	`Prints results of last parse to current printer`
`'q' or 'quit'`	`Quits interface`
`'ref'`	`Refreshes the screen`
`'last'`	`Parses last discourse again`

Among the commands, two particularly important ones are 'quit' (to quit the program) and 'set' (to set various parameters of the system). 'help ¡command¿' will display online info about that command.

The Prolog Interpreter

The `prolog` interpreter[16] is started by typing 'loop.'. The interpreter is similar to the `perl` interpreter in that it accepts both sentences and commands. The

[16]This code is derived from a program written for FraCaS by Steve Pulman.

conventions for multiple sentences texts are similar to those used in the `perl` interpreter. The currently accepted commands are ‘q’ (for quit), ‘h’ (for ‘help’), and ‘ptq’ (to run a test suite; currently being revised).

Grammars and Grammar Switching

EKNTOOL comes with three grammars, `fracas_main`, `fracas_undersp`, and `stdrt`. It is possible to use any of those; the default is `fracas_main`. The default can be changed by setting the value of the environment variable `EKNTOOLGRAMMAR`.

To switch grammar from within the `perl` interpreter, do:

```
   set grammar <grammar_name>
```

where `<grammar_name>` is one of the accepted grammars (currently).

From `prolog`, do:

```
?-  load_grammar(<grammar_name>).
```

Writing Permissions

In order to run the program you need writing permission in the directory from which the program is invoked, since ekntool creates various files and directories for its output. You also need to be able to overwrite the files ”stdrt.tex”, ”done”, and ”disouts/file1” in your current directory.

2.3.3 The Main Grammar

The `fracas_main` grammar covers basic anaphora, quantification, negation, and tense. It can deal with simple discourses. Here are some of the sentences handled by the grammar:

```
john walks
smith walked.
mary arrived
a man walks
every representative ran
he leaves
```

```
every representative of a company left
a representative of every company arrived
a representative of nlpcom ran
every representative that owned a computer left
a representative that owned every workstation arrived
a company that owned every computer of nlpcom left
smith hired jones
a company likes robin
every customer likes smith
john owns a workstation
john hired every consultant
john hired every consultant who owns a computer
a man likes a woman
every customer likes a computer
every man likes every woman
john likes a man that likes bill
john likes a man that likes him
a company that liked john hired him
john likes a man that likes him
a man likes a man that likes him
a man likes every man that likes him
john doesnt walk
john does not walk
john didnt walk
john did not walk
john does not like a computer
every company that owns a computer likes it
every farmer likes a donkey who likes him
john does not like a man that likes him
john does not like a man that doesnt like him
john likes a man who does not like him.
a man that liked a man that liked john hired him
john walks. he talks
a man walks. he talks.
every man walks. he talks.
john likes mary. she likes him
every man owns a workstation. he likes it.
every farmer who owns a donkey beats it. john beats it
anna doesnt own a workstation. she likes it.
if bill walks then he talks.
if a man walks then he talks
if every man walks then he talks
if a man likes bill then he hires him
if every man loves bill then he beats him
if a man likes a woman then he loves her
if a farmer owns a donkey then he beats it
if john likes bill then bill talks
if bill walks then he talks. he runs.
if a man walks then he talks. he runs.
```

2.3.4 Installation

The Files

The release 0.4 of EKNTOOL consists of the following files:

- A README file.
- the `perl` scripts `ekntool` and `showekn_perl`
- the `sicstus` source files:
  1. the load files `load.pl`, `load_pred.pl` and `main.pl`;
  2. the files containing programs for operations on EKN expressions: `ekn.pl`, `ekn_ops.pl`, `traverse.pl`;
  3. the code for the conversion between `prolog` and LaTeX, in `display.pl` and `showekn.pl`;
  4. the code for the interface between `perl` and `prolog` in `perl_prolog.pl`;
  5. various auxiliary files, in `extras.pl`, `mp_util.pl`, `utils.pl`
  6. the grammars, in the subdirectory `grammars`. Each grammar `<grammar_name>` is in a directory with the same name. All grammars include a file called `load.pl` called by EKNTOOL, a file called `lexicon.pl`, possibly a file called `db` that contains the test suite for the grammar, and additional files containing the actual code.
- the latex style files `ekn.sty`, which contains the definitions of the macros used to draw DRT-style boxes.
- the PLN database plndb .

The code is version-controlled using rcs; most of the source files are in the RCS directory, in ",v" form. They can be extracted using the "co" command.

Porting the System to a New Host

To move the code to another directory or system, you have to fix the `ekntool` perl script and the `load_pred.pl` file. In particular, you have to

1. modify the variables `$EKNTOOLHOME` and `$EKNTOOLLATEX` in the `perl` script.

2. in `load_pred.pl`, you have to fix the definition of the predicate `ekntool_home`.

Both of these changes are done automatically by editing the Makefile file in the top directory and replacing the name of the variable `$EKNTOOLHOME`, then typing 'make' (see below).

The following programs are used by ekntool:

- sicstus prolog: it is invoked by calling "sicstus".
- latex: it is invoked by calling "latex".
- xdvi: it is invoked by calling "xdvi".

Check before using the system that all of these programs are accessible.

Installing the System

ekntool comes in a tar file. Save the tar file in the directory where you want the code to be located, and type

```
tar -xf <tar file name>
```

this will create a subdirectory called `ekntool` which, in turn, will have a subdirectory called `grammars`. Change directory to the directory `ekntool`, and edit the file `Makefile` changing the value of the variable `EKNTOOLHOME` to the value returned by typing `pwd` in the ekntool directory; then type

```
make
```

This will compile all the files, including the grammar files.

Common Problems

The prolog files should be compiled. This should take place automatically by running 'make' after untarring the files.

When making a change to one of the grammars, make sure to recompile the file before starting 'ekntool' again. This is especially true if the file to be modified is the ‘lexicon.pl’ file. If you add a new lexical entry for ‘pavarotti’ and also update the word list at the beginning of that file, ekntool will think that the word ‘pavarotti’ is in the lexicon, and will call prolog. The lack of a lexical item is especially bad because it makes the system backtrack to death, so you will get in this case a particularly unilluminating ‘Sicstus Error!’ message.

The `perl` interpreter and the prolog interpreter are kept synchronized by having the former wait for prolog to create a file called ‘done’ in the directory in which ekntool is executed. If prolog enters a non-terminating loop, the `perl` interpreter hangs waiting for the file being created. The same may happen if latex breaks when processing the input. In order to see what prolog or latex are doing, set the debug switch to 3. (See online help.)

When processing a sentence, the `perl` interpreter checks if all words are in the lexicon before sending the sentence to prolog. The interpreter expects to find the list of words in a file called `lexicon.pl` in the directory containing the current grammar. The list of words should be on a line initiated by `%??`. A common problem is to add a word to the lexicon, but not to this list of words.

2.3.5 Writing a New Grammar

It is possible to add new grammars to the system. In order to add a new grammar, you have to create a subdirectory of the directory `$EKNTOOLHOME/grammars`. This directory must contain a file called `load.pl` that will load the files that define the grammar, as well as a file called `lexicon.pl` that contains at the beginning a specification of the lexicon. (This is only necessary if using the `perl` interpreter; see below.) The only constraint on the grammar is that it must define a three-places predicate

```
top(X,M,H)
```

where `X` is the input sentence, `M` is the result of semantic interpretation in EKN format as defined in `$EKNTOOLHOME/ekn.pl`, and `H` is the history of the derivation (to be used by the system when the ‘verbose’ option is set). The predicate `top` will be invoked repeatedly by ekntool until all solutions have been found.

The input to the grammar is a list of words. The current convention is for periods (‘.’) in the natural language input to be converted into slashes (‘/’); thus, the sentence ”Smith runs.” will result in top being invoked with first argument `[smith,runs,/]`.

The history list should be a list of the form:

```
<hl> ::= [<component>+]
```

(note that it cannot be an empty list!) whereas each component is of the form:

```
<component> ::= [<tag>,M,{<component>,<hl>}+]
```

The 'tag' can be anything you want. We use two different kinds of tags: one for syntactic categories, the other for semantic operations. The system doesn't interpret those, it just prints them in bold font.

To use the new grammar from prolog it is sufficient to add a directory for the new grammar. In order to switch to the new grammar from ekntool (i.e., in order to be able to do

```
set grammar <grammar_name>
```

from the `perl` interpreter) it will also be necessary to modify the procedure 'switch' in the ekntool code. switch first checks that the value of the switch is one of those allowed, then sets the parameters accordingly. In order to add a new grammar, it is necessary to add the grammar name to the list of values allowed.

A grammar may include a 'test suite' contained in a test file. The test suite for a grammar should be put in a file called `db` in that grammar's directory. The file should consist of a list of sentences, one for line. The number of readings for a sentence may also be specified at the end of the sentence, separated by a `<TAB>` from the end of the sentence. (See the db file in `$EKNTOOLHOME` for an example.) If the directory for a grammar includes a file called `db`, the `perl` interpreter loads that file and uses the test suite when the command `test` is used.

2.3.6 EKN in Prolog

Here is a brief description of how the EKN notation is converted into `prolog` notation.

We use `prolog` variables to represent EKN parameter constants and `prolog` constants to represent all other constants.[17] Literals (the basic INFONS of Situation

[17]Parameter constants are implemented as prolog variables to take advantage of unification for doing substitutions in e.g. β-conversion. However, parameter constants could also be implemented as prolog constants (as in one of the β-conversion tools included in the framework library).

Theory) are represented by prolog terms of the form infon(Rel,Args,Pol) where Rel is a term denoting a relation, Args is a list of terms and Pol is the 'polarity' of the literal (0 or 1). For example, the literal $\mathbf{meets}(X, Y)$ is represented by the prolog term infon(meets,[X,Y],1).

Infons can be conjoined or disjoined. We use terms of the form /\[P,Q]to indicate the cojunction of the infons P and Q, and terms of the form \/[P,Q] to represent their disjunction.

Propositions are represented by terms of the form S /= I, where S is a term denoting a situation and I is a code denoting an infon (which may be formed by conjoining or disjoining other infons). Propositions can also be conjoined and disjoined. Propositions can also be negated: the negation of a proposition P is represented by the prolog term ~P.

We represent assignments as either

- a prolog list whose elements are infixed terms of the form i>X where i is an index and X is a prolog term representing an EKN expression, or
- a prolog list whose elements are prolog terms representing EKN expressions.

An example of assignment of the first type is [i>X,j>Y]; of the second type, [X,Y]. Expressions of the second type are interpreted as representing assignments whose domain is the sequence of natural numbers from 1 up to the number of elements in the list. For example, [X,Y] is taken to be equivalent to [1>X,2>Y].

If A is a prolog term representing an assignment and X a prolog term representing an arbitrary EKN expression, then A^^X is the prolog term that represents an abstract. For example, the abstract represented in EKN as in (2.1a) is represented by the prolog term in (2.1b).

(2.1) a. $\lambda \left[i \rightarrow \mathrm{X} \quad j \rightarrow \mathrm{Y} \right] . \mathbf{meets}(X, Y)$

b. [i>X,j>Y]^^infon(meets,[X,Y],1)

An application is represented by the prolog term X*Y, where X is a prolog term denoting an abstract and Y a prolog term denoting an assignment.

Restricted objects are represented by prolog terms of the form X/Y, where X is any prolog term representing an EKN expression and Y is a proposition (not necessarily atomic).

2.3.7 Troubleshooting

When starting the `perl` interpreter, the following sequence of events should occur:

1. the screen is cleared
2. 'Starting sicstus process' appears on the screen
3. 'Getting current lexicon from lexicon.pl' appears
4. The prompt appears.

If all this works, i.e., if you get the prompt ok, then *before* typing a sentence, you should type `set debug 3`. In this way, you'll get the reports from `sicstus` and from LaTeX.

Chapter 3

A Semantic Test Suite

How can the semantic competence of NLP systems be adequately measured? The question is important both from a practical and a theoretical perspective. Its practical importance derives from an increasingly felt need for fair and accurate means of comparing the quality of competing language processing systems — e.g. for purposes of funding, or of assessing the possibilities for commercial development — but also as a yardstick for the developers of such systems themselves. Test suites are theoretically important insofar as they force those who design them to think their way through to a clear conception of the semantic capacities that various types of NLP systems should have. In particular, we believe that test suite designers should have a well-developed idea of a core competence that ought to be common to all NLP systems that pretend to a genuine semantic component. That there should be such a core, that it should be substantial, and that it should go well beyond the semantic components of any NLP system currently running, is something of which we are firmly convinced.

In the light of the view expressed elsewhere in this and other FraCaS deliverables (see for instance section 4.(a).iii.A in Deliverable 16) that inferential ability is not only a central manifestation of semantic competence but is in fact centrally constitutive of it, it shouldn't be a surprise surprise that we regard inferencing tasks as the best way of testing an NLP system's semantic capacity. Such tests take the following form: the system is given some natural language input T, then is presented with a claim S and asked to determine whether S follows from T. Since tests should be neutral as regards the semantic representation formalisms they use, the test claim S should of course also be offered in the form of a natural language sentence.

Tasks of this sort form a continuum. At one end we find cases where the input T consists of a small number of sentences (sometimes only one) which are directly relevant to the test sentence S, much in the style of problems in introductory logic books of the type: "Formalize the argument below, then derive the conclusion from the premises or else construct a counterexample". Tasks at the other end of the spectrum rather resemble the problems one often finds in books that are used in the kind of "text analysis" that is nowadays often taught in high schools: the student is presented with a text and then with certain questions about it. The questions vary, but some at least are yes-no questions the answer to which can be determined by straight logical deduction from what the text actually says, questions such as: "Did so-and-so do such-and such before the new president was elected?", "Is person x related to person y?", etc. On the whole the questions one finds in such tests are not quite as simple-minded as those one would want to put to an NLP system in order to assess its logic.

But even when the questions they present are quite simple, tasks of this type tend to be non-trivial insofar as they require searching through a large quantity of potential premises and discarding all that is irrelevant. This is something with which human beings seem to have comparatively little difficulty, but which seriously affects the efficiency of NLP systems with the kind of architecture that most current systems have. Premise search is a problem that will be familiar to anyone with some knowledge of theorem proving from large data bases.

While the difference between the two types of inferential tasks we have just mentioned is obviously a matter of degree, it is a difference that should be firmly kept in mind when perusing the list of "tests" given below. Some of the items on this list may seem very elementary indeed, and the ability to solve them in the form in which they are presented here — i.e. as tasks of the first kind — should perhaps not be counted for very much. But it is quite a different thing to be able to confirm or reject inferences of the exemplified patterns from substantial bits of text.

The complexity of this second kind of task grows not only with the size of the premise set, but also with the variety of inference principles that the system has at its disposition. Already a theorem prover that includes all inference schemata instantiated in our list faces a serious complexity problem; but of course, this is only a small sample from the totality of inference patterns that a competent natural language theorem prover should have at its command.

An additional complicating factor is that a fair test suite should contain not only valid inferences (i.e. not only queries of the form "Does S follow from the given input T?" to which the answer is "yes"), but also invalid ones. Otherwise it would be vulnerable to conscious and unconscious cheating. But when can the system be certain that S does not follow from T; how can it distinguish between cases where no proof of S from T exist and cases where it hasn't yet managed to find such a proof? We know that there can be no complete answer to this question, for first order logic is undecidable and the expressive power of natural language includes that of first order logic (and in fact goes well beyond it).

One could avoid this last problem somewhat by designing test suites for which it is guaranteed that the queried conclusion S either follows from T or else is refutable from it. This finesses the decision problem and might be recommendable at least as a first start. But in the long run we should probably be more demanding. It seems to be part of the logical competence of the human language user that he can not only see of many propositions that they follow from a given input, but also that he can see of countless others that they do not follow. We suspect that this latter ability is no less important in the semantic processing of natural language than the ability to see that many things do follow. If so, then NLP systems should be equipped with this facility too, and in that case it would be fair to test them for it.

Although these introductory remarks have been very sketchy and left a number of knots untied, we hope they have made one thing clear: The list below is not only incomplete in that the great majority of inference patterns are not represented; it is also non-specific in that it leaves open precisely what a test should look like that is based on the patterns that are represented.

The inferences are grouped into linguistic and semantic phenomena loosely following the structure of the FraCaS deliverable **D7:ch.3 Some Basic Linguistic Data and Their Importance** and are presented in the form of a list of premisses followed by a query followed by an answer and in some cases comments. Unless indicated otherwise the premisses constitute the only source of information available for the inference.

3.1 Generalized quantifiers

3.1.1 Conservativity

Q As are Bs $\leftrightarrow$ Q As are As who are Bs

(3.1) An Italian became the world's greatest tenor.

Was there an Italian who became the world's greatest tenor?

[Yes]

(3.2) Every Italian man wants to be a great tenor.
Some Italian men are great tenors.

Are there Italian men who want to be a great tenor?

[Yes]

(3.3) All Italian men want to be a great tenor.
Some Italian men are great tenors.

Are there Italian men who want to be a great tenor?
[Yes]

(3.4) Each Italian tenor wants to be great.
Some Italian tenors are great.

Are there Italian tenors who want to be great?
[Yes]

(3.5) The really ambitious tenors are Italian.

Are there really ambitious tenors who are Italian?
[Yes]

(3.6) No really great tenors are modest.

Are there really great tenors who are modest?
[No]

(3.7) Some great tenors are Swedish.

Are there great tenors who are Swedish?
[Yes]

(3.8) Many great tenors are German.

Are there great tenors who are German?
[Yes]

(3.9) Several great tenors are British.

Are there great tenors who are British?
[Yes]

(3.10) Most great tenors are Italian.

Are there great tenors who are Italian?
[Yes]

(3.11) A few great tenors sing popular music.
Some great tenors like popular music.

Are there great tenors who sing popular music?
[Yes]

(3.12) Few great tenors are poor.

Are there great tenors who are poor?

[Not many]

(3.13) Both leading tenors are excellent.
Leading tenors who are excellent are indispensable.

Are both leading tenors indispensable?

[Yes]

(3.14) Neither leading tenor comes cheap.
One of the leading tenors is Pavarotti.

Is Pavarotti a leading tenor who comes cheap?

[No]

(3.15) At least three tenors will take part in the concert.

Are there tenors who will take part in the concert?

[Yes]

(3.16) At most two tenors will contribute their fees to charity.

Are there tenors who will contribute their fees to charity?

[At most two]

3.1.2 Monotonicity (upwards on second argument)

Q As are Bs and all Bs are Cs $\rightarrow$ Q As are Cs

(3.17) An Irishman won the Nobel prize for literature.

Did an Irishman win a Nobel prize?

[Yes]

(3.18) Every European has the right to live in Europe.
Every European is a person.
Every person who has the right to live in Europe
can travel freely within Europe.

Can every European travel freely within Europe?

[Yes]

(3.19) All Europeans have the right to live in Europe.
Every European is a person.
Every person who has the right to live in Europe
can travel freely within Europe.

Can all Europeans travel freely within Europe?
[Yes]

(3.20) Each European has the right to live in Europe.
Every European is a person.
Every person who has the right to live in Europe
can travel freely within Europe.

Can each European travel freely within Europe?
[Yes]

(3.21) The residents of member states have the right to live in Europe.
All residents of member states are individuals.
Every individual who has the right to live in Europe
can travel freely within Europe.

Can the residents of member states travel freely within Europe?
[Yes]

(3.22) No delegate finished the report on time.

Did no delegate finish the report?
[Don't know]

(3.23) Some delegates finished the survey on time.

Did some delegates finish the survey?
[Yes]

(3.24) Many delegates obtained interesting results from the survey.

Did many delegates obtain results from the survey?
[Yes]

(3.25) Several delegates got the results published
in major national newspapers.

Did several delegates get the results published?
[Yes]

(3.26) Most Europeans are resident in Europe.
All Europeans are people.
All people who are resident in Europe
can travel freely within Europe.

Can most Europeans travel freely within Europe?
[Yes]

(3.27) A few committee members are from Sweden.
All committee members are people.
All people who are from Sweden are from Scandinavia.

Are at least a few committee members from Scandinavia?
[Yes]

(3.28) Few committee members are from Portugal.
All committee members are people.
All people who are from Portugal are from southern Europe.

Are there few committee members from southern Europe?
[Don't know]

(3.29) Both commissioners used to be leading businessmen.

Did both commissioners used to be businessmen?
[Yes]

(3.30) Neither commissioner spends a lot of time at home.

Does neither commissioner spend time at home?
[Don't know]

(3.31) A least three commissioners spend a lot of time at home.

Do at least three commissioners spend time at home?
[Yes]

(3.32) At most ten commissioners spend a lot of time at home.

Do at most ten commissioners spend time at home?
[Don't know]

3.1.3 Monotonicity (downwards on second argument)

Q As are Bs and all Cs are Bs $\rightarrow$ Q As are Cs

(3.33) An Irishman won a Nobel prize.

Did an Irishman win the Nobel prize for literature?

[Don't know]

(3.34) Every European can travel freely within Europe.
Every European is a person.
Every person who has the right to live in Europe
can travel freely within Europe.

Does every European have the right to live in Europe?

[Don't know]

(3.35) All Europeans can travel freely within Europe.
Every European is a person.
Every person who has the right to live in Europe
can travel freely within Europe.

Do all Europeans have the right to live in Europe?

[Don't know]

(3.36) Each European can travel freely within Europe.
Every European is a person.
Every person who has the right to live in Europe
can travel freely within Europe.

Does each European have the right to live in Europe?

[Don't know]

(3.37) The residents of member states can travel freely within Europe.
All residents of member states are individuals.
Every individual who has the right to live anywhere in Europe
can travel freely within Europe.

Do the residents of member states have the right to live
anywhere in Europe?

[Don't know]

(3.38) No delegate finished the report.

Did any delegate finish the report on time?

[No]

(3.39) Some delegates finished the survey.

Did some delegates finish the survey on time?

[Don't know]

(3.40) Many delegates obtained results from the survey.

Did many delegates obtain interesting results from the survey?

[Don't know]

(3.41) Several delegates got the results published.

Did several delegates get the results published
in major national newspapers?

[Don't know]

(3.42) Most Europeans can travel freely within Europe.
All Europeans are people.
All people who are resident in Europe
can travel freely within Europe.

Are most Europeans resident in Europe?

[Don't know]

(3.43) A few committee members are from Scandinavia.
All committee members are people.
All people who are from Sweden are from Scandinavia.

Are at least a few committee members from Sweden?

[Don't know]

(3.44) Few committee members are from southern Europe.
All committee members are people.
All people who are from Portugal are from southern Europe.

Are there few committee members from Portugal?

[Yes]

(3.45) Both commissioners used to be businessmen.

Did both commissioners used to be leading businessmen?

[Don't know]

(3.46) Neither commissioner spends time at home.

Does either commissioner spend a lot of time at home?

[No]

(3.47) A least three commissioners spend time at home.

Do at least three commissioners spend a lot of time at home?

[Don't know]

(3.48) At most ten commissioners spend time at home.

Do at most ten commissioners spend a lot of time at home?

[Yes]

3.1.4 Monotonicity (upwards on first argument)

Q As are Bs and all As are Cs $\rightarrow$ Q Cs are Bs

(3.49) A Swede won a Nobel prize.
Every Swede is a Scandinavian.

Did a Scandinavian win a Nobel prize?

[Yes]

(3.50) Every Canadian resident can travel freely within Europe.
Every Canadian resident is a resident of
the North American continent.

Can every resident of the North American continent
travel freely within Europe?

[Don't know]

(3.51) All Canadian residents can travel freely within Europe.
Every Canadian resident is a resident of
the North American continent.

Can all residents of the North American continent
travel freely within Europe?

[Don't know]

(3.52) Each Canadian resident can travel freely within Europe.
Every Canadian resident is a resident of
the North American continent.

Can each resident of the North American continent
travel freely within Europe?

[Don't know]

(3.53) The residents of major western countries can travel freely within Europe.
All residents of major western countries are residents of western countries.

Do the residents of western countries have the right to live in Europe?

[Don't know]

(3.54) No Scandinavian delegate finished the report on time.

Did any delegate finish the report on time?

[Don't know]

(3.55) Some Irish delegates finished the survey on time.

Did any delegates finish the survey on time?

[Yes]

(3.56)

Many British delegates obtained interesting results from the survey.

Did many delegates obtain interesting results from the survey?

[Don't know]

(3.57) Several Portuguese delegates got the results published in major national newspapers.

Did several delegates get the results published in major national newspapers?

[Yes]

(3.58) Most Europeans who are resident in Europe can travel freely within Europe.

Can most Europeans travel freely within Europe?

[Don't know]

(3.59) A few female committee members are from Scandinavia.

Are at least a few committee members from Scandinavia?

[Yes]

(3.60) Few female committee members are from southern Europe.

Are few committee members from southern Europe?

[Don't know]

(3.61) Both female commissioners used to be in business.

Did both commissioners used to be in business?

[Yes, if both commissioners are female.
Otherwise there are more than two commissioners]

(3.62) Neither female commissioner spends a lot of time at home.

Does either commissioner spend a lot of time at home?

[No, if both commissioners are female.
Otherwise there are more than two commissioners]

(3.63) A least three female commissioners spend time at home.

Do at least three commissioners spend time at home?

[Yes]

(3.64) At most ten female commissioners spend time at home.

Do at most ten commissioners spend time at home?

[Don't know]

3.1.5 Monotonicity (downwards on first argument)

Q As are Bs and all Cs are As $\rightarrow$ Q Cs are Bs

(3.65) A Scandinavian won a Nobel prize.
Every Swede is a Scandinavian

Did a Swede win a Nobel prize?

[Don't know]

(3.66) Every resident of the North American continent can travel freely within Europe.
Every Canadian resident is a resident of the North American continent.

Can every Canadian resident freely within Europe?

[Yes]

(3.67) All residents of the North American continent can travel freely within Europe.
Every Canadian resident is a resident of the North American continent.

Can all Canadian residents travel freely within Europe?
[Yes]

(3.68) Each resident of the North American continent can travel freely within Europe.
Every Canadian resident is a resident of the North American continent.

Can each Canadian resident travel freely within Europe?
[Yes]

(3.69) The residents of western countries can travel freely within Europe.
All residents of major western countries are residents of western countries.

Do the residents of major western countries have the right to live in Europe?
[Yes]

(3.70) No delegate finished the report on time.

Did any Scandinavian delegate finish the report on time?
[No]

(3.71) Some delegates finished the survey on time.

Did any Irish delegates finish the survey on time?
[Don't know]

(3.72)
Many delegates obtained interesting results from the survey.

Did many British delegates obtain interesting results from the survey?
[Don't know]

(3.73)
Several delegates got the results published in major national newspapers.

Did several Portuguese delegates get the results published in major national newspapers?
[Don't know]

(3.74) Most Europeans can travel freely within Europe.
Can most Europeans who are resident outside Europe travel freely within Europe?
[Don't know]

(3.75) A few committee members are from Scandinavia.
Are at least a few female committee members from Scandinavia?
[Don't know]

(3.76) Few committee members are from southern Europe.
Are few female committee members from southern Europe?
[Yes]

(3.77) Both commissioners used to be in business.
Did both female commissioners used to be in business?

[Yes, if both commissioners are female.
Otherwise there are more than two commissioners]

(3.78) Neither commissioner spends a lot of time at home.
Does either female commissioner spend a lot of time at home?

[No, if both commissioners are female.
Otherwise there are more than two commissioners]

(3.79) A least three commissioners spend time at home.
Do at least three male commissioners spend time at home?
[Don't know]

(3.80) At most ten commissioners spend time at home.
Do at most ten female commissioners spend time at home?
[Yes]

3.2 Plurals

A number of inferences pertaining to plurals are covered under the headings of generalized quantifiers and elsewhere. Here we concentrate on conjoined NPs; bare, existential and definite plurals; dependent plurals; and collective and distributive readings and scope ambiguity.

3.2.1 Conjoined Noun Phrases

(3.81) Smith, Jones and Anderson signed the contract.
Did Jones sign the contract?
[Yes]

(3.82) Smith, Jones and several lawyers signed the contract.
Did Jones sign the contract?
[Yes]

(3.83) Either Smith, Jones or Anderson signed the contract.
Did Jones sign the contract?
[Don't know]

(3.84) Either Smith, Jones or Anderson signed the contract.
If Smith and Anderson did not sign the contract,
did Jones sign the contract?
[Yes]

(3.85) Exactly two lawyers and three accountants signed the contract.
Did six lawyers sign the contract?
[No]
No scope relations between the two conjoined NPs

(3.86) Exactly two lawyers and three accountants signed the contract.
Did six accountants sign the contract?
[No]
No scope relations between the two conjoined NPs

Conjoined Nbars

Nbar conjunction tends to be quite ambiguous. This may be the result of a syntactic ambiguity between (i) genuine Nbar conjunction, and (ii) NP conjunction where the determiner of one of the NPs is ellided.

(3.87) Every representative and client was at the meeting.
Was every representative at the meeting?
[Yes, on one reading]
Arguably NP conjunction: every representative and every client

(3.88) Every representative and client was at the meeting.
Was every representative at the meeting?
[Don't know, on one reading]
NBar conjunction: everyone who is both a representative and a client

(3.89) Every representative or client was at the meeting.
Was every representative and every client at the meeting?
[Yes]
With disjunction, NP conjunction seems unavailable.

3.2.2 Definite Plurals

Definite plurals can often be non-anaphoric and behave like universally quantified noun phrases (3.90). However, as with (generic) bare plurals, the force of the quantification can also be less than universal (3.91). Whether this lessening of quantificational force is due to the noun phrase or to the predicate of which the NP is an argument is unclear (3.92,3.93).

(3.90) The chairman read out the items on the agenda.
Did the chairman read out every item on the agenda?
[Yes]
Non-anaphoric, universal plural definite

(3.91) The people who were at the meeting voted for a new chairman.
Did everyone at the meeting vote for a new chairman?
[Don't know]
Some people may have abstained from the vote

(3.92) All the people who were at the meeting voted for a new chairman.
Did everyone at the meeting vote for a new chairman?
[Yes]

(3.93) The people who were at the meeting all voted for a new chairman.
―――
Did everyone at the meeting vote for a new chairman?
[Yes]

Closely related to this, plural definites can have a collective/institutional or even generic interpretation:

(3.94) The inhabitants of Cambridge voted for a Labour MP.
―――
Did every inhabitant of Cambridge vote for a Labour MP?
[Don't know]

(3.95) The Ancient Greeks were noted philosophers.
―――
Was every Ancient Greek a noted philosopher?
[Don't know]

(3.96) The Ancient Greeks were all noted philosophers.
―――
Was every Ancient Greek a noted philosopher?
[Yes]

3.2.3 Bare Plurals

Bare plurals can exhibit existential, (quasi) universal, generic or dependent plural behaviour. The circumstances giving rise to these different behaviours a poorly understood, so we only give a few illustrative examples.

(3.97) Software faults were blamed for the system failure.
―――
Was the system failure blamed on one or more software faults?
[Yes]
Existential bare plural

(3.98) Software faults were blamed for the system failure.
Bug # 32-985 is a known software fault.
―――
Was Bug # 32-985 blamed for the system failure?
[Don't know]
Existential interpretation: not every software fault contributed to the

failure.

(3.99)

Clients at the demonstration were all impressed by the system's performance.
Smith was a client at the demonstration.

Was Smith impressed by the system's performance?

[Yes]

(Quasi) universal bare plural

(3.100)

Clients at the demonstration were impressed by the system's performance.

Were most clients at the demonstration impressed by the system's performance?

[Yes]

(Quasi) universal bare plural

(3.101) University graduates make poor stock-market traders.
Smith is a university graduate.

Is Smith likely to make a poor stock market trader?

[Yes]

Generic interpretation

(3.102) University graduates make poor stock-market traders.
Smith is a university graduate.

Will Smith make a poor stock market trader?

[Don't know]

Generic interpretation

3.2.4 Dependent Plurals

(3.103) All APCOM managers have company cars.
Jones is an APCOM manager.

Does Jones have a company car?

[Yes]

(3.104) All APCOM managers have company cars.
Jones is an APCOM manager.

Does Jones have more than one company car?
[Don't know]

3.2.5 Negated Plurals

(3.105) Just one accountant attended the meeting.

Did no accountants attend the meeting?
[No]

(3.106) Just one accountant attended the meeting.

Did no accountant attend the meeting?
[No]

(3.107) Just one accountant attended the meeting.

Did any accountants attend the meeting?
[Yes]

(3.108) Just one accountant attended the meeting.

Did any accountant attend the meeting?
[Yes]

(3.109) Just one accountant attended the meeting.

Did some accountants attend the meeting?
[No/Just one]

(3.110) Just one accountant attended the meeting.

Did some accountant attend the meeting?
[Yes]

3.2.6 Collective and Distributive Plurals

(3.111) Smith signed one contract.
Jones signed another contract.

Did Smith and Jones sign two contracts?
[Yes, on a collective/cumulative
reading of the conclusion]

(3.112) Smith signed two contracts.
Jones signed two contracts.

Did Smith and Jones sign two contracts?
[Yes, on a distributive reading
of "Smith and Jones"]

(3.113) Smith signed two contracts.
Jones also signed them.

Did Smith and Jones sign two contracts?
[Yes]

3.3 (Nominal) Anaphora

In the examples below we make the assumption (unless otherwise indicated) that there is no context beyond that provided by the mini-discourse. This is so that we can do away with explicit co-indexing of pronouns with their antecedents, on the grounds that context will provide only (or sometimes a strictly limited number) of possible antecedents.

3.3.1 Intra-Sentential

(3.114) Mary used her workstation.

Was Mary's workstation used?
[Yes]

(3.115) Mary used her workstation.

Does Mary have a workstation?
[Yes]

(3.116) Mary used her workstation.

Is Mary female?
[Yes]

(3.117) Every student used her workstation.
Mary is a student.

Did Mary use her workstation?
[Yes]

(3.118) Every student used her workstation.
Mary is a student.

Does Mary have a workstation?
[Yes]

(3.119) No student used her workstation.
Mary is a student.

Did Mary use a workstation?
[No]

3.3.2 Inter-Sentential

(3.120) Smith attended a meeting.
She chaired it.

Did Smith chair a meeting?
[Yes]

(3.121)

Smith delivered a report to ITEL.
She also delivered them an invoice.
And she delivered them a project proposal.

Did Smith deliver a report, an invoice and a project proposal to ITEL?
[Yes]

Keeping track of same entities across more than two sentences.

(3.122)

Every committee has a chairman.
He is appointed its members.

Does every committee have a chairman appointed by members of the committee?
[Yes]

Modal subordination.

3.3.3 Plural Anaphora

(3.123) ITEL has sent most of the reports Smith needs.
They are on her desk.

Are there some reports from ITEL on Smith's desk?
[Yes]

(3.124) Two out of ten machines are missing.
They have been removed.

Have two machines been removed?
[Yes]

(3.125) Two out of ten machines are missing.
They have been removed.

Have eight machines been removed?
[Don't know]

Set difference can't be used to construct plural antecedents

(3.126) Two out of ten machines are missing.
They were all here yesterday.

Were ten machines here yesterday?
[Yes]

(3.127)
Smith took a machine on Tuesday, and Jones took a machine on Wednesday.
They put them in the lobby.

Did Smith and Jones put two machines in the lobby?
[Yes, on a distributive reading of the question]

Construction of plural antecedents from separate constituents.

(3.128) John and his colleagues went to a meeting.
They hated it

Did John's colleagues hate the meeting?
[Yes]

(3.129) John and his colleagues went to a meeting.
They hated it

Did John hate the meeting?
[Yes, on one possible reading]
"They" can be resolved to John and his colleagues

(3.130) John and his colleagues went to a meeting.
They hated it

Did John hate the meeting?
[Don't know, on one possible reading]
"They" can also be resolved to John's colleagues but not John

(3.131) Each department has a dedicated line.
They rent them from BT.

Does every department rent a line from BT?
[Yes]
Dependent plural anaphora

(3.132) Each department has a dedicated line.
The sales department rents it from BT.

Does the sales department rent a line from BT?
[Yes]
Paycheque pronoun

3.3.4 E-type and Donkey Pronouns

(3.133) GFI owns several computers.
ITEL maintains them.

Does ITEL maintain all the computers that GFI owns
[Yes]
E-type anaphora

(3.134)
Every customer who owns a computer has a service contract for it.
MFI is a customer that owns exactly one computer.

Does MFI have a service contract for all its computers
[Yes]

Donkey sentence

(3.135)

Every customer who owns a computer has a service contract for it.
MFI is a customer that owns several computers.

Does MFI have a service contract for all its computers

[Yes]

This pattern of inference, unlike (3.134), tends to some theory dependence. Although the inference seems correct in this example, compare with (3.136)

(3.136)

Every executive who had a laptop computer brought it to take notes at the meeting.
Smith is a executive who owns five different laptop computers.

Did Smith take five laptop computers to the meeting?

[Don't know]

Similar to (3.135), except for tense and pragmatic plausibility.

(3.137)

There are 100 companies.
ICM is one of the companies and owns 150 computers.
It does not have service contracts for any of its computers.
Each of the other 99 companies owns one computer.
They have service contracts for them.

Do most companies that own a computer have a service contract for it?

[Yes]

Proportion problem

3.3.5 Functional and Subsectional

Due to the heavy domain dependence of functional (or better perhaps, relational) anaphora, it is hard to state general inferences that don't assume considerable background knowledge unless this is given explicitly.

(3.138) Every report has a cover page.
R-95-103 is a report.
Smith signed the cover page.
Did Smith sign the cover page of R-95-103?
[Yes]

3.3.6 Simple Reflexives

(3.139) A company director awarded himself a large payrise.
Has a company director awarded and been awarded a payrise?
[Yes]

(3.140) John said Bill had hurt himself.
Did John say Bill had been hurt?
[Yes]

(3.141) John said Bill had hurt himself.
Did anyone say John had been hurt?
[Don't know]

3.4 Ellipsis

In nearly all cases the inferences presented here have conclusions that are simply reconstructions of the ellided constituent. Unfortunately, an inference test suite is not well suited to illustrating prohibitions on ellipsis resolution. For example, an ill-formed discourse like

John was in Paris yesterday. *So did Bill.

doesn't even get as far as supporting any inferences.

3.4.1 VP Ellipsis

(3.142) John spoke to Mary.
So did Bill

Did Bill speak to Mary?
[Yes]

Basic example.

(3.143) John spoke to Mary.
So did Bill
John spoke to Mary at four o'clock.

Did Bill speak to Mary at four o'clock?
[Don't know]

Temporal resolution of tense in antecedent is not carried across to ellipsis.

(3.144) John spoke to Mary at four o'clock.
So did Bill

Did Bill speak to Mary at four o'clock?
[Yes]

Explicit temporal adverbials are carried across to ellipsis.

(3.145) John spoke to Mary at four o'clock.
And Bill did at five o'clock

Did Bill speak to Mary at five o'clock?
[Yes]

Explicit temporal adverbials are not carried across if overridden.

(3.146) John has spoken to Mary.
Bill is going to.

Will Bill speak to Mary?
[Yes]

Tense agreement not necessary between ellipsis and antecedent.

(3.147) John spoke to Mary on Monday.
Bill didn't.

Did Bill speak to Mary on Monday?
[No]

Polarity agreement not necessary between ellipsis and antecedent.

(3.148) Has John spoken to Mary?
Bill has.

Has Bill spoken to Mary?
[Yes]

Mood agreement not necessary between ellipsis and antecedent.

(3.149) John has spoken to Mary.
The students have too.

Have the students spoken to Mary?
[Yes]

Number agreement not necessary.

3.4.2 Gapping

(3.150) John went to Paris by car, and Bill by train.

Did Bill go to Paris by train?
[Yes]

Basic example

(3.151) John went to Paris by car, and Bill by train to Berlin.

Did Bill go to Berlin by train?
[Yes]

Another basic example

(3.152) John went to Paris by car, and Bill to Berlin.

Did Bill go to Berlin by car?
[Yes]

Another basic example

(3.153) John is going to Paris by car, and the students by train.

Are the students going to Paris by train?

[Yes]

Subject-verb agreement not necessary

(3.154) John went to Paris by car.

Bill by train.

Did Bill go to Paris by train?

[Yes]

Cross-sentential gapping

3.4.3 One Anaphora

(3.155) John owns a car.

Bill owns one too.

Does Bill own a car?

[Yes]

Basic example

(3.156) John owns a car.

Bill owns one too.

Is there a car that John and Bill own?

[Don't know]

It needn't be the same car that John and Bill own.

(3.157) John owns a red car.

Bill owns a blue one.

Does Bill own a blue car?

[Yes]

(3.158) John owns a red car.

Bill owns a blue one.

Does Bill own a red car?

[Don't know]

(3.159) John owns a red car.
Bill owns a fast one.

Does Bill own a fast car?
[Yes]

(3.160) John owns a red car.
Bill owns a fast one.

Does Bill own a fast red car?
[Yes, on one possible reading]
The one anaphor may be resolved via the property of being a red car

(3.161) John owns a red car.
Bill owns a fast one.

Does Bill own a fast red car?
[Don't know, on one possible reading]
Or the one anaphor may just be resolved via the property of being a car

(3.162) John owns a fast red car.
Bill owns a slow one.

Does Bill own a slow red car?
[Yes]
When semantically parallel (e.g. fast/slow) modifiers appear on the antecedent and one-anaphor, it appears that all non-parallel modifiers must form part of the resolution

3.4.4 Sluicing

(3.163) John had his paper accepted.
Bill doesn't know why.

Does Bill know why John had his paper accepted?
[No]

3.4.5 Phrasal Ellipsis

(3.164) John spoke to Mary.
And to Sue.

Did John speak to Sue?
[Yes]
PP ellipsis (subcategorized)

(3.165) John spoke to Mary.
On Friday.

Did John speak to Mary on Friday?
[Yes]
PP ellipsis: adds PP to antecedent

(3.166) John spoke to Mary on Thursday.
And on Friday.

Did John speak to Mary on Friday?
[Yes]
PP ellipsis: replaces PP in antecedent

(3.167) Twenty men work in the Sales Department.
But only one woman.

Do two women work in the Sales Department?
[No]
NP ellipsis

(3.168) Five men work part time.
And forty five women.

Do forty five women work part time?
[Yes]
NP ellipsis

(3.169) John found Mary before Bill.

Did John find Mary before Bill found Mary?
[Yes, on one possible reading]
NP ellipsis

(3.170) John found Mary before Bill.
Did John find Mary before John found Bill?
[Yes, on one possible reading]
NP ellipsis

(3.171) John wants to know how many men work part time.
And women.
Does John want to know how many women work part time?
[Yes]
Nbar ellipsis

(3.172) John wants to know how many men work part time, and which.
Does John want to know which men work part time?
[Yes]
Determiner ellipsis

3.4.6 Antecedent Contained Deletion

Antecedent contained deletion is a notorious problem for copying approaches to ellipsis, since the antecedent clause contains the ellipsis and some way must be found of removing it from the copy.

(3.173) Bill spoke to everyone that John did.
John spoke to Mary.
Did Bill speak to Mary?
[Yes]

(3.174) Bill spoke to everyone that John did.
Bill spoke to Mary.
Did John speak to Mary?
[Don't know]

3.4.7 Configurational Effects

There are a number of syntactic and other configurational constraints on what can constitute the antecedent to an ellipsis. These constraints varying depending on the type of ellipsis (VP, phrasal, gapping, etc).

(3.175) John said Mary wrote a report, and Bill did too.

Did Bill say Mary wrote a report?

[Yes, on one possible reading/parse]

(3.176) John said Mary wrote a report, and Bill did too.

Did John say Bill wrote a report?

[Yes, on one possible reading/parse]

(3.177) John said that Mary wrote a report, and that Bill did too.

Did Bill say Mary wrote a report?

[Don't know]

Note that the first sentence in (3.175) and (3.176) is syntactically ambiguous, depending on whether the conjunctive clause conjoins with the main or subordinate clause of *John said Mary wrote a report.* In (3.177) the conjunctive clause unambiguously conjoins with the subordinate clause, and only one interpretation of the ellipsis is possible. This appears to indicate that the antecedent clause to a VP ellipsis must be adjacent to the elliptical clause. However, as the examples below show, this is not correct.

(3.178) John wrote a report, and Bill said Peter did too.

Did Bill say Peter wrote a report?

[Yes]

Embedded elliptical clause

(3.179) If John wrote a report, then Bill did too.
John wrote a report.

Did Bill write a report?

[Yes]

Elliptical and antecedent clause embedded (in parallel)

(3.180) John wanted to buy a car, and he did.

Did John buy a car?

[Yes]

Embedded antecedent clause

(3.181) John needed to buy a car, and Bill did.

Did Bill buy a car?

[Don't know]

Other configurational effects of the kinds illustrated in Deliverable 7 are hard to exemplify using inference suites.

3.4.8 Ellipsis and Anaphora

The following inferences illustrate differences between strict and sloppy interpretations of anaphors in elliptical clauses

(3.182) Smith represents his company and so does Jones.

Does Jones represent Jones' company?

[Yes, on one reading]

Sloppy identity

(3.183) Smith represents his company and so does Jones.

Does Jones represent Smith's company?

[Yes, on one reading]

Strict identity

(3.184) Smith represents his company and so does Jones.

Does Smith represent Jones' company?

[Don't know]

(3.185) Smith claimed he had costed his proposal and so did Jones.

Did Jones claim he had costed his own proposal?

[Yes, on one reading]

Sloppy identity on both pronouns

(3.186) Smith claimed he had costed his proposal and so did Jones.

Did Jones claim he had costed Smith's proposal?

[Yes, on one reading]

Sloppy identity "he", strict on "his"

(3.187) Smith claimed he had costed his proposal and so did Jones.

Did Jones claim Smith had costed Smith's proposal?

[Yes, on one reading]

Strict identity on both pronouns

(3.188) Smith claimed he had costed his proposal and so did Jones.

Did Jones claim Smith had costed Jones' proposal?

[Don't know]

Can't have strict identity on "he" and sloppy identity on "his"

(3.189) John is a man and Mary is a woman.
John represents his company and so does Mary.

Does Mary represent her own company?

[Yes, on one reading]

Sloppy identity, gender agreement not necessary

(3.190) John is a man and Mary is a woman.
John represents his company and so does Mary.

Does Mary represent John's company?

[Yes, on one reading]

Strict identity, gender agreement not necessary

(3.191)

Bill suggested to Frank's boss that they should go to the meeting together, and Carl to Alan's wife.

If it was suggested that Bill and Frank should go together, was it suggested that Carl and Alan should go together?

[Yes]

Plural pronouns resolved in parallel

(3.192)

Bill suggested to Frank's boss that they should go to the meeting together, and Carl to Alan's wife.

If it was suggested that Bill and Frank should go together,
was it suggested that Carl and Alan's wife should go together?

[Don't know]

Plural pronouns resolved in parallel

(3.193)

Bill suggested to Frank's boss that they should go to the meeting together, and Carl to Alan's wife.

If it was suggested that Bill and Frank's boss should go together,
was it suggested that Carl and Alan's wife should go together?

[Yes]

Plural pronouns resolved in parallel

(3.194)

Bill suggested to Frank's boss that they should go to the meeting together, and Carl to Alan's wife.

If it was suggested that Bill and Frank's boss should go together,
was it suggested that Carl and Alan should go together?

[Don't know]

Plural pronouns resolved in parallel

(3.195)

Bill suggested to Frank's boss that they should go to the meeting together, and Carl to Alan's wife.

If it was suggested that Bill, Frank and Frank's boss should go together,
was it suggested that Carl, Alan and Alan's wife should go together?

[Yes]

Plural pronouns resolved in parallel

3.4.9 Ellipsis and Quantification

Scope parallelism turns out to be rather tricky to illustrate through inference suites. This is because of the entailment relation: $\exists\forall \models \forall\exists$.

(3.196) A lawyer signed every report, and so did an auditor.
That is, there was one lawyer who signed all the reports.

Was there one auditor who signed all the reports?
[Yes]

3.5 Adjectives

The inferences below carve up adjectives into (a by no means exhaustive) cross-cutting set of dimensions. Typical inferences are given for example adjectives.

3.5.1 Affirmative and Non-Affirmative

Affirmative adjectives map the denotation of the predicate they modify onto a subset of the denotation. So for example, an old man is a man. Most adjectives are affirmative, but a few like *former* and *fake* are not. Given that someone is a former student, one cannot conclude that they are now a student. But it is not entirely clear whether one can conclude that they are not now a student — they may have become one again.

(3.197) John has a genuine diamond.

Does John have a diamond?
[Yes]
Affirmative adjectives: Adj N $\Rightarrow$ *N*

(3.198) John is a former university student.

Is John a university student?
[No/Don't know]
Non-affirmative: Adj N $\not\Rightarrow$ *N*
(Opinions differ about whether "Adj N $\Rightarrow \neg$*N")*

(3.199) John is a successful former university student.

Is John successful?
[Yes (for a former university student)]
Ordering between affirmative and non-affirmative adjectives affects which adjectival predications are and aren't affirmed

(3.200) John is a former successful university student.

Is John successful?

[Don't know]

(3.201) John is a former successful university student.

Is John a university student?

[Don't know]

John may currently be an unsuccessful university student

3.5.2 No Comparison Class

Gradable adjectives (e.g. *big, small*) usually assume some form of comparison class (i.e. 'big for an N'). But some others do not e.g. *four-legged*, or the adjectival phrase *ten foot long*. Adjectives not requiring a comparison class permit straightforward predication without reference to a nominal property providing a comparison class: a ten foot long alligator is ten foot long.

(3.202) Every mammal is an animal.

Is every four-legged mammal a four-legged animal?

[Yes]

$[N1 \rightarrow N2] \Rightarrow [Adj(N1) \rightarrow Adj(N2)]$

(3.203) Dumbo is a four-legged animal.

Is Dumbo four-legged?

[Yes]

$Adj(N)(x) \Rightarrow Adj(x)$

3.5.3 Opposites

Large and *small* (applied to the same comparison class) are opposites. If something is a small N it cannot be a large N, and vice versa. Some things can be neither large nor small Ns.

(3.204) Mickey is a small animal.

Is Mickey a large animal?

[No]

$Small(N) \Rightarrow \neg Large(N)$

(3.205) Dumbo is a large animal.

Is Dumbo a small animal?

[No]

Large(N) $\Rightarrow$ $\neg$*Small(N)*

(3.206) Fido is not a small animal.

Is Fido a large animal?

[Don't know]

$\neg$*Small(N)* $\not\Rightarrow$ *Large(N)*

(3.207) Fido is not a large animal.

Is Fido a small animal?

[Don't know]

$\neg$*Large(N)* $\not\Rightarrow$ *Small(N)*

(3.208) Mickey is a small animal.
Dumbo is a large animal

Is Mickey smaller than Dumbo?

[Yes]

"Small" and "large" are related via the comparative "smaller"

(3.209) Mickey is a small animal.
Dumbo is a large animal

Is Mickey larger than Dumbo?

[No]

"Small" and "large" are related via the comparative "larger"

3.5.4 Extensional Comparison Classes

Adjectives like *large* and *small* depend only on the extension of the comparison class they depend on.

(3.210) All mice are small animals.
Mickey is a large mouse.

Is Mickey a large animal?

[No]

(3.211) All elephants are large animals.
Dumbo is a small elephant.

Is Dumbo a small animal?
[No]

(3.212) All mice are small animals.
All elephants are large animals.
Mickey is a large mouse.
Dumbo is a small elephant.

Is Dumbo larger than Mickey?
[Yes]
Assume comparative relations exemplified in (3.208) and (3.209)

(3.213) All mice are small animals.
Mickey is a large mouse.

Is Mickey small?
[??: Yes for a mouse
?? No for an animal]
Adjectives requiring a comparison class cannot usually be predicated in the absence of a common noun, unless some comparison class is clear from the wider context.

3.5.5 Extensional and Intensional Comparison Classes

Some adjectives require an 'intensional' comparison class: different inferences may follow when two distinct but co-extensive predicates provide the comparison class.

(3.214) All legal authorities are law lecturers.
All law lecturers are legal authorities.

Are all fat legal authorities fat law lecturers?
[Yes]
Extensional comparison class

(3.215) All legal authorities are law lecturers.
All law lecturers are legal authorities.
Are all competent legal authorities competent law lecturers?
[Don't know]
Intensional comparison class

(3.216) John is a fatter politician than Bill.
Is John fatter than Bill?
[Yes]
Extensional

(3.217) John is a cleverer politician than Bill.
Is John cleverer than Bill?
[Don't know]
Intensional

Note that both intensional and extensional comparison class adjectives support comparatives.

3.5.6 Default Comparison Classes

Comparison class adjectives can sometimes pick up a default comparison class from the subject NP. For example, knowing that Kim is a person provides a default scale for assessing cleverness in people. If Kim were known to be a dog, the assessment scale would be different.

(3.218) Kim is a clever person.
Is Kim clever?
[Yes]

(3.219) Kim is a clever politician.
Is Kim clever?
[Don't know]

3.6 Comparatives

3.6.1 Phrasal Comparatives

(3.220) The PC-6082 is faster than the ITEL-XZ.
The ITEL-XZ is fast.
___
Is the PC-6082 fast?
[Yes]

(3.221) The PC-6082 is faster than the ITEL-XZ.
___
Is the PC-6082 fast?
[Don't know]

(3.222) The PC-6082 is faster than the ITEL-XZ.
The PC-6082 is fast.
___
Is the ITEL-XZ fast?
[Don't know]

(3.223) The PC-6082 is faster than the ITEL-XZ.
The PC-6082 is slow.
___
Is the ITEL-XZ fast?
[No]

(3.224) The PC-6082 is as fast as the ITEL-XZ.
The ITEL-XZ is fast.
___
Is the PC-6082 fast?
[Yes]

(3.225) The PC-6082 is as fast as the ITEL-XZ.
___
Is the PC-6082 fast?
[Don't know]

(3.226) The PC-6082 is as fast as the ITEL-XZ.
The PC-6082 is fast.
___
Is the ITEL-XZ fast?
[Don't know]

(3.227) The PC-6082 is as fast as the ITEL-XZ.
The PC-6082 is slow.

Is the ITEL-XZ fast?
[No]

(3.228) The PC-6082 is as fast as the ITEL-XZ.

Is the PC-6082 faster than the ITEL-XZ?
[Don't know]

(3.229) The PC-6082 is as fast as the ITEL-XZ.

Is the PC-6082 slower than the ITEL-XZ?
[No]

(3.230) ITEL won more orders than APCOM did.

Did ITEL win some orders?
[Yes]

(3.231) ITEL won more orders than APCOM did.

Did APCOM win some orders?
[Don't know]

(3.232) ITEL won more orders than APCOM did.
APCOM won ten orders.

Did ITEL win at least eleven orders?
[Yes]

Inferences (3.233)–(3.235) are similar to (3.230)–(3.232). Note however, that if "APCOM" can be interpreted as referring to a particular order (e.g. "the APCOM contract"), as it can in (3.233), the sentence *ITEL won more orders than APCOM* is ambiguous between a reading like that in (3.230)–(3.232), and one where ITEL won more than just the APCOM order — see (3.236)

(3.233) ITEL won more orders than APCOM.

Did ITEL win some orders?
[Yes]

(3.234) ITEL won more orders than APCOM.

Did APCOM win some orders?
[Don't know]

(3.235) ITEL won more orders than APCOM.
APCOM won ten orders.

Did ITEL win at least eleven orders?
[Yes]

(3.236) ITEL won more orders than the APCOM contract.

Did ITEL win the APCOM contract?
[Yes]

(3.237) ITEL won more orders than the APCOM contract.

Did ITEL win more than one order?
[Yes]

(3.238) ITEL won twice as many orders than APCOM.
APCOM won ten orders

Did ITEL win twenty orders?
[Yes]

3.6.2 Clausal Complement

(3.239) ITEL won more orders than APCOM lost.

Did ITEL win some orders?
[Yes]

(3.240) ITEL won more orders than APCOM lost.

Did APCOM lose some orders?
[Don't know]

(3.241) ITEL won more orders than APCOM lost.
APCOM lost ten orders.

Did ITEL win at least eleven orders?
[Yes]

3.6.3 Measure Phrases

(3.242) The PC-6082 is faster than 500 MIPS.
The ITEL-ZX is slower than 500 MIPS

Is the PC-6082 faster than the ITEL-ZX?
[Yes]

3.6.4 Differential Comparatives

(3.243) ITEL sold 3000 more computers than APCOM.
APCOM sold exactly 2500 computers.

Did ITEL sell 5500 computers?
[Yes]

3.6.5 Attributive Comparatives

(3.244) APCOM has a more important customer than ITEL.

Does APCOM have a more important customer than ITEL is?
[Yes, on one reading of the premise]

(3.245) APCOM has a more important customer than ITEL.

Does APCOM has a more important customer than ITEL has?
[Yes, on one reading of the premise]

3.6.6 Comparatives and Quantifiers

(3.246) The PC-6082 is faster than every ITEL computer.
The ITEL-ZX is an ITEL computer.

Is the PC-6082 faster than the ITEL-ZX?
[Yes]

(3.247) The PC-6082 is faster than some ITEL computer.
The ITEL-ZX is an ITEL computer.

Is the PC-6082 faster than the ITEL-ZX?
[Don't know]

(3.248) The PC-6082 is faster than any ITEL computer.
The ITEL-ZX is an ITEL computer.

Is the PC-6082 faster than the ITEL-ZX?
[Yes]

(3.249) The PC-6082 is faster than the ITEL-ZX and the ITEL-ZY.

Is the PC-6082 faster than the ITEL-ZX?
[Yes]

(3.250) The PC-6082 is faster than the ITEL-ZX or the ITEL-ZY.

Is the PC-6082 faster than the ITEL-ZX?
[Yes, on one reading of the premise]

3.7 Temporal Reference

Inference patterns involving temporal reference are complicated by the interplay between tense, aspectual information, lexical semantics, defeasible interpretation principles such as narrative progression, rhetorical relations, a theory of action and causation, world knowledge, interaction between plurality, genericity and temporal/aspectual phenomena etc. Some of the inferences are very basic, some are more involved. The more complex examples give ample illustration of the fact that temporal phenomena are usually discourse phenomena.

3.7.1 Standard Use of Tenses

(3.251) ITEL has a factory in Birmingham.

Does ITEL currently have a factory in Birmingham?
[Yes]

(3.252) Since 1992 ITEL has been in Birmingham.
It is now 1996.

Was ITEL in Birmingham in 1993?
[Yes]

(3.251) and (3.252) are instances of the subinterval property. This works only with stative verbs. C.f. the following example involving an accomplishment verb in the simple past:

(3.253) ITEL has developed a new editor since 1992.
It is now 1996

Did ITEL develop a new editor in 1993?
[Don't know]

Similarly with activity verbs and adverbial modification

(3.254) ITEL has expanded since 1992.
It is now 1996.

Did ITEL expand in 1993?
[Don't know]

Also, the position of the "since" adverbial affects the range of readings available:

(3.255) Since 1992 ITEL has made a loss.
It is now 1996.

Did ITEL make a loss in 1993?
[Yes]

(3.256) ITEL has made a loss since 1992.
It is now 1996.

Did ITEL make a loss in 1993?
[Don't know, on one reading of the premise]

(3.257) ITEL has made a loss since 1992.
It is now 1996.

Did ITEL make a loss in 1993?
[Yes, on one reading of the premise]

(3.258) In March 1993 APCOM founded ITEL.

Did ITEL exist in 1992?
[No]

(3.258) involves the lexical semantics of *found.*

3.7.2 Temporal Adverbials

Indexicals

Non-context dependent indexicals are reasonably straightforward:

(3.259) The conference started on July 4th, 1994.
It lasted 2 days.
___
Was the conference over on July 8th, 1994?
[Yes]

Context dependent indexicals (e.g. *today*, *yesterday*) are evaluated with respect to some temporal reference point (e.g. *now*):

(3.260) Yesterday APCOM signed the contract.
Today is Saturday, July 14th.
___
Did APCOM sign the contract Friday, 13th.?
[Yes]

'Before', 'After' (Temporal Subordinate Clauses)

Ignoring counterfactual readings, *before* and *after* have the following transitivity properties: if X, Y and Z are either all state or accomplishment or achievement or activity denoting sentences we have

(3.261)
$$\begin{array}{c} X \rhd Y. \\ Y \rhd Z. \\ X \rhd Z. \end{array}$$
where $\rhd \in \{\text{before}, \text{after}\}$

(3.262) Smith left after Jones left.
Jones left after Anderson left.
___
Did Smith leave after Anderson left?
[Yes]

In general transitivity does not hold when we mix aspectual classes in the premises:

(3.263) Smith was present after Jones left.
Jones left after Anderson was present.
Was Smith present after Anderson was present?
[Don't know]

If X and Y are either all accomplishment or achievement denoting sentences with simple tenses *before* and *after* are inverses of each other:

(3.264) X before Y iff Y after X.

(3.265) Smith left.
Jones left.
Smith left after Jones left.
Did Jones leave before Smith left?
[Yes]

(3.266) Smith left.
Jones left.
Jones left before Smith left.
Did Smith leave after Jones left?
[Yes]

(3.267) Jones revised the contract.
Smith revised the contract.
Jones revised the contract before Smith did.
Did Smith revise the contract after Jones did.
[Yes]

(3.268) Jones revised the contract.
Smith revised the contract.
Jones revised the contract after Smith did.
Did Smith revise the contract before Jones did.
[Yes]

In general this is not so with activity verbs:

(3.269) Smith swam.
Jones swam.
Smith swam before Jones swam.
Did Jones swim after Smith swam?
[Don't know]

However we do get

(3.270) Smith swam to the shore.
Jones swam to the shore.
Smith swam to the shore before Jones swam to the shore.
———
Did Jones swim to the shore after Smith swam to the shore?
[Yes]

Here the PP *to the shore* provides an end point or conclusion for the activity.

Before and *after* are not inverses for state denoting sentences:

(3.271) Smith was present.
Jones was present.
Smith was present after Jones was present.
———
Was Jones present before Smith was present?
[Don't know]

(3.272) Smith was present.
Jones was present.
Smith was present before Jones was present.
———
Was Jones present after Smith was present?
[Don't know]

(3.273) Smith was writing a report.
Jones was writing a report.
Smith was writing a report before Jones was writing a report.
———
Was Jones writing a report after Smith was writing a report.?
[Don't know]

(3.274) Smith was writing a report.
Jones was writing a report.
Smith was writing a report after Jones was writing a report.
———
Was Jones writing a report before Smith was writing a report?
[Don't know]

Also *before*, but not *after*, can have a counterfactual meaning. Whether this is a distinct sense of *before* is open to debate:

(3.275) Smith left the meeting before he lost his temper.
Did Smith lose his temper?
[Don't know]

With *when* things are even more complicated. The problem is that it is often very difficult to tease apart the temporal from the causal dimension of *when*, c.f.

(3.276) When they opened the M25, traffic increased.

'In', 'For' and 'On' Temporal Adverbials

In and *for* adverbials can be used as tests for the aspectual class of verb phrases (or sentences).

(3.277) Smith lived in Birmingham in 1991.
Did Smith live in Birmingham in 1992?
[Don't know]
Stative

(3.278) Smith wrote his first novel in 1991.
Did Smith write his first novel in 1992?
[No]
(Unrepeatable) accomplishment

(3.279) Smith wrote a novel in 1991.
Did Smith write it in 1992?
[No]
(Unrepeatable) accomplishment

(3.280) Smith wrote a novel in 1991.
Did Smith write a novel in 1992?
[Don't know]
(Repeatable) accomplishment

(3.281) Smith was running a business in 1991.
Was Smith running it in 1992?
[Don't know]
Activity

(3.282) Smith discovered a new species in 1991.
Did Smith discover it in 1992?
[No]
(Unrepeatable) achievement

(3.283) Smith discovered a new species in 1991.
Did Smith discover a new species in 1992?
[Don't know]
(Repeatable) achievement

(3.284) Smith wrote a report in two hours.
Smith started writing the report at 8 am.
Had Smith finished writing the report by 11 am?
[Yes]
Accomplishment

(3.285) Smith wrote a report in two hours.
Did Smith spend two hours writing the report?
[Don't know]
Smith may have written the report in less than two hours. It is unclear whether there are two different readings for the premise: one where Smith takes exactly two hours, and one where he does it within two hours.

(3.286) Smith wrote a report in two hours.
Did Smith spend more than two hours writing the report?
[No]

(3.287) Smith wrote a report in two hours.
Did Smith write a report in one hour?
[Don't know]

(3.288) Smith wrote a report in two hours.

Did Smith write a report?

[Yes]

(3.289) Smith discovered a new species in two hours.

Did Smith spend two hours discovering the new species

[No]

Achievements are typically (more or less) instantaneous

(3.290) Smith discovered a new species in two hours.

Did Smith discover a new species

[Yes]

(3.291) Smith discovered many new species in two hours.

Did Smith spend two hours discovering new species

[?Yes]

Repeated achievement can last two hours

(3.292) Smith was running his own business in two years.

Did Smith spend two years running his own business?

[Don't know]

Premise refers to time taken to inception of activity, not duration of activity.

(3.293) Smith was running his own business in two years.

Did Smith spend more than two years running his own business?

[Don't know]

Cf similar inference for accomplishment, (3.286)

(3.294) Smith was running his own business in two years.

Did Smith run his own business?

[Yes]

(3.295) In two years Smith owned a chain of businesses.

Did Smith own a chain of business for two years?

[Don't know]

States behave like activities.

(3.296) In two years Smith owned a chain of businesses.

Did Smith own a chain of business for more than two years?

[Don't know]

(3.297) In two years Smith owned a chain of businesses.

Did Smith own a chain of business?

[Yes]

(3.298) Smith lived in Birmingham for two years.

Did Smith live in Birmingham for a year?

[Yes]

State

(3.299) Smith lived in Birmingham for two years.

Did Smith live in Birmingham for exactly a year?

[No]

(3.300) Smith lived in Birmingham for two years.

Did Smith live in Birmingham?

[Yes]

(3.301) Smith ran his own business for two years.

Did Smith run his own business for a year?

[Yes]

Activity

(3.302) Smith ran his own business for two years.

Did Smith run his own business?

[Yes]

(3.303) Smith wrote a report for two hours.

Did Smith write a report for an hour?

[Yes]

Accomplishment

(3.304) Smith wrote a report for two hours.

Did Smith write a report?

[Don't know]

He may not have finished it

(3.305) #Smith discovered a new species for an hour.

(3.306) Smith discovered new species for two years.

Did Smith discover new species?

[Yes]

Repeated achievement

Quantificational Adverbials

(3.307) In 1994 ITEL sent a progress report every month.

Did ITEL send a progress report in July 1994?

[Yes]

Quantificational adverbials also introduce scope ambiguities with respect to other quantified NPs

(3.308) Smith wrote to a representative every week.

Is there a representative that Smith wrote to every week?

[Yes on one scoping; don't know on another scoping]

3.7.3 Anaphoric Dimension

Rhetorical relations like narrative progression are defeasible interpretation principles. They depend on a theory of action and causation and general world knowledge (c.f. (3.309) and (3.310)).

(3.309) Smith left the house at a quarter past five.
She took a taxi to the station
and caught the first train to Luxembourg.

(3.310) Smith lost some files.
They were destroyed when her hard disk crashed.

(3.311) Smith had left the house at a quarter past five.
Then she took a taxi to the station.

Did Smith leave the house before she took a taxi to the station
[Yes]

3.7.4 Adverbs of Quantification

(3.312) ITEL always delivers reports late.
In 1993 ITEL delivered reports.

Did ITEL delivered reports late in 1993?
[Yes]

(3.313) ITEL never delivers reports late.
In 1993 ITEL delivered reports.

Did ITEL delivered reports late in 1993?
[No]

3.7.5 Some more Complex Examples

(3.314) Smith arrived in Paris on the 5th of May, 1995.
Today is the 15th of May, 1995.
She is still in Paris.

Was Smith in Paris on the 7th of May, 1995?
[Yes]

(3.315)
When Smith arrived in Katmandu
she had been travelling for three days.

Had Smith been travelling the day before she arrived in Katmandu?
[Yes]

(3.316) Jones graduated in March and has been employed ever since.
Jones has been unemployed in the past.

Was Jones unemployed at some time before he graduated?
[Yes]

(3.317)
Every representative has read this report.
No two representatives have read it at the same time.
No representative took less than half a day to read the report.
There are sixteen representatives.

Did it take the representatives more than a week to read the report?
[Yes]

(3.318) While Jones was updating the program,
Mary came in and told him about the board meeting.
She finished before he did.

Did Mary's story last as long as Jones's updating the program?
[No]

(3.319)
Before APCOM bought its present office building,
it had been paying mortgage interest on the previous one for 8 years.
Since APCOM bought its present office building
it has been paying mortgage interest on it for more than 10 years.

Has APCOM been paying mortgage interest
for a total of 15 years or more?
[Yes]

(3.320) When Jones got his job at the CIA,
he knew that he would never be allowed to write his memoirs.

Is it the case that Jones is not and
will never be allowed to write his memoirs?
[Yes]

(3.321) Smith has been to Florence twice in the past.
Smith will go to Florence twice in the coming year.

Two years from now will Smith have been to Florence
at least four times?
[Yes]

(3.322) Last week I already knew that when, in a months time,
Smith would discover that she had been duped
she would be furious.

Will it be the case that in a few weeks Smith will discover
that she has been duped; and will she be furious?

[Yes]

(3.323) No one gambling seriously stops until he is broke.
No one can gamble when he is broke.

Does everyone who starts gambling seriously stop
the moment he is broke?

[Yes]

(3.324) No one who starts gambling seriously stops until he is broke.

Does everyone who starts gambling seriously continue
until he is broke?

[Yes]

(3.325) Nobody who is asleep ever knows that he is asleep.
But some people know that they have been asleep
after they have been asleep.

Do some people discover that they have been asleep?

[Yes]

3.8 Verbs

3.8.1 Aspectual Classes

See also the inference pertaining to *in* and *for* adverbials.

(3.326) ITEL built MTALK in 1993.

Did ITEL finish MTALK in 1993?

[Yes]

(3.327) ITEL was building MTALK in 1993.

Did ITEL finish MTALK in 1993?

[Don't know]

(3.328) ITEL won the contract from APCOM in 1993.
Did ITEL win a contract in 1993?
[Yes]

(3.329) ITEL was winning the contract from APCOM in 1993.
Did ITEL win a contract in 1993?
[Don't know]

(3.330) ITEL owned APCOM from 1988 to 1992.
Did ITEL own APCOM in 1990?
[Yes]

3.8.2 Distributive and Collective Predication

(3.331) Smith and Jones left the meeting.
Did Smith leave the meeting
[Yes]

(3.332) Smith and Jones left the meeting.
Did Jones leave the meeting
[Yes]

(3.333) Smith, Anderson and Jones met.
Was there a group of people that met?
[Yes]

3.9 Attitudes

3.9.1 Epistemic, Intentional and Reportive Attitudes

(3.334) Smith knew that ITEL had won the contract in 1992.
Did ITEL win the contract in 1992?
[Yes]

(3.335) Smith believed / said / denied / feared / hoped that ITEL had won the contract in 1992.
Did ITEL win the contract in 1992?
[Don't know]

(3.336) ITEL managed to win the contract in 1992.
Did ITEL win the contract in 1992?
[Yes]

(3.337) ITEL tried /wanted to win the contract in 1992.
Did ITEL win the contract in 1992?
[Don't know]

(3.338) It is true that ITEL won the contract in 1992.
Did ITEL win the contract in 1992?
[Yes]

(3.339) It is false that ITEL won the contract in 1992.
Did ITEL win the contract in 1992?
[No]

3.9.2 Preceptive Attitudes: "See" with Bare Infinitive Complements

Inferences we do not get

(3.340) Smith saw Jones sign the contract
If Jones signed the contract, his heart was beating
Did Smith see Jones' heart beat?
[Don't know]

(3.341) Smith saw Jones sign the contract
When Jones signed the contract, his heart was beating
Did Smith see Jones' heart beat?
[Don't know]

Veridicality

a saw $\phi \rightarrow \phi$

(3.342) Smith saw Jones sign the contract
$\overline{}$
Did Jones sign the contract?
[Yes]

Substitution

a saw $\phi(b)$, $b = c \rightarrow a$ saw $\phi(c)$

(3.343) Smith saw Jones sign the contract
Jones is the chairman of ITEL
$\overline{}$
Did Smith see the chairman of ITEL sign the contract?
[Yes]

Existential instantiation

a saw $\phi(b) \rightarrow \exists x$ a saw $\phi(x)$

(3.344) Helen saw the chairman of the department answer the phone
The chairman of the department is a person
$\overline{}$
Is there anyone whom Helen saw answer the phone?
[Yes]

Conjunction distribution

a saw $\phi \wedge \psi \rightarrow a$ saw ϕ and a saw ψ

(3.345) Smith saw Jones sign the contract and his secretary make a copy
$\overline{}$
Did Smith see Jones sign the contract?
[Yes]

Disjunction distribution

a saw $\phi \vee \psi \rightarrow a$ saw ϕ or a saw ψ

(3.346) Smith saw Jones sign the contract or cross out the crucial clause

---

Did Smith either see Jones sign the contract or see Jones cross out the crucial clause?

[Yes]

Chapter 4

Computational Lexical Semantics in Review

This chapter was contributed by Ted Briscoe, University of Cambridge Computer Laboratory, ejb@cl.cam.ac.uk

4.1 Introduction

In this short overview, whose title I have shamelessly adapted from Beth Levin's [1995] influential review of linguistic work on (verbal) lexical semantics, I try to rereview this and related work in the light of the last decade's research and also to identify developments relevant to computational lexical semantics. In the final section, I summarise how the Acquilex (Esprit Basic Research) project has attempted to extend and exploit this work in the development of a prototype computational lexicon incorporating lexical semantic information which integrates with computational approaches to proof/model-theoretic compositional semantics (of the type explored in the FraCaS (CEC LRE) project).

The study of lexical semantics and of the lexicon has seen a resurgence of activity during the last decade. Hudson [1995] convincingly argues that linguistic theory provided little justification for the then prevalent mainstream assumptions that:

1. the lexicon is a distinct component of the grammar,
2. the lexicon is a discrete list of lexical entries, and
3. the lexicon contains only intralinguistic information.

Assumption (2) has been largely abandoned and replaced by a conception of the lexicon as an inheritance network incorporating at least is-a links (e.g. [Pollard and Sag, 1987]) and possibly default inheritance (e.g. [Evans and Gazdar, 1989b]). One important implication of this development for lexical semantics is the possibility of treating the polysemy/homonymy distinction more satisfactorily by treating polysemy in terms of such links rather than ignoring it and enumerating all word senses as a list of distinct lexical entries (e.g. [Pustejovsky and Boguraev, 1993]). Assumption (1) is the subject of much more focussed debate as a result of the work on network-based lexical knowledge representation languages (LKRLs): in particular, Gazdar & Evans propose that whilst syntax and compositional semantics can be expressed within a monotonic unification-based formalism, lexical processes and relations require a more expressive non-monotonic formalism such as their path equation LKRL, DATR; whilst, within HPSG, it has been proposed that the lexicon is represented as a set of typed feature structures related by subsumption, effectively reducing the lexicon to the syntactic/semantic formalism. However, the status of lexical rules within the HPSG framework has been contentious and problematic (e.g. [Riehemann, 1993; Calcagno, 1995]). Assumption (3) has received less attention, but nevertheless can be considered in a more focussed manner in the light of the advent of LKRLs. Both DATR and the HPSG lexicon are couched in formalisms which allow only a proper subset of the operations defined in general purpose KRLs which have been proposed within the AI literature. It remains to be seen whether lexical relations and processes can be adequately expressed within these more restricted formalisms.

These brief remarks, I hope make it clear that lexical (semantic) work has progressed somewhat beyond the situation which Hudson criticises and which he exemplifies with the GPSG ([Gazdar *et al.*, 1985]) lexical entry for *weep* repeated below:

```
<weep,
 [[-N,[+V],[BAR 0],[SUBCAT 1]],
 {wept},
 WEEP'>
```

in which one sense and syntactic realisation is represented by a minimal description of orthography/phonology, syntactic (sub)category, irregular morphological variant form, and unanalysed semantic primitive, as part of a large list of unrelated other entries. In what follows I attempt to flesh out these developments in more detail. I will review lexical (semantic) research under the four subheadings of *argument structure*, *event structure*, *qualia structure* and *inheritance structure*. These are borrowed from Pustejovsky [1991; 1995], whose work has done much to rekindle and refocus work on lexical semantics. I don't attempt to be at all comprehensive either in coverage of active topics of research or in citing relevant work: rather, I attempt to identify the important ideas and give main references from which further work can be found.

4.1.1 Argument Structure

Work on argument structure, defined as a specification of the number, syntactic and semantic type, and syntactic realisation of a predicate's arguments, has been the focus of most generative research on lexical semantics and particularly on the syntactico-semantic interface. Levin [1995] exclusively reviews such work which primarily addresses the related problems of linking (capturing generalisations concerning the syntactic realisation of different arguments) and of diathesis alternations (the same argument may be realised differently or omitted). For example, in *John broke the vase* the verb *break* appears with two arguments, an agent and a theme / affected object. In general, agents (if realised) are subjects and themes are objects, as here. However, as *break* can undergo the causative/inchoative alternation, the theme can be realised as subject *The vase broke.* These facts are seen as lying at the syntactico-semantic interface since they appear to be motivated by aspects of the lexical semantics of the predicate. For instance, *break* and *eat* both undergo (in)transitivity alternations (*John ate (chips)*) but only *break* undergoes causative/inchoative because it is a unaccusative change-of-state verb. Frequently, alternations modify the semantic class or entailments of a predicate, for example *slide* is extended from a change-of-position to a change-of-possession verb when it undergoes the dative alternation – *John slid the computer to the door / John slid *the door/Bill the computer.* Such phenomena and treatments of it have been exhaustively surveyed in [Levin, 1993].

Attempts to express rules of linking and alternation have either been couched in terms of case/semantic/thematic roles (e.g. [Fillmore, 1968]) or in terms of predicate decomposition (e.g. [Carter, 1976]). One problem with both approaches from the perspective of computational lexical semantics is that usually neither approach is related to a proof-theoretic framework. One exception is the work of Dowty [1989] which treats thematic roles as bundles of entailments prototypically associated with predicates when the associated argument is realised. This captures both the fact that alternations typically affect meaning and that the entailments associated with a thematic role, such as agent, are not uniform across all predicates (e.g. volitionality *The drunk collided with the lamppost / The drunk wanted a scotch*). It also supports a prototypic semantic account of linking in which, for example, the argument associated with the greatest number of prototypically agentive entailments is realised as subject. Dowty modifies Davidson's event-based semantic framework to include proto-thematic roles as predicates relating event variables to participants denoted by arguments. The representation of *John broke the vase* would be roughly: $\exists(e,x) \wedge\ vase(x) \wedge\ break(e) \wedge\ p\_agt(e,j) \wedge p\_pat(e,x)$.

4.1.2 Event Structure

Event structure refers to the (Vendlerian) classification of predicates into states, processes and transitions (achievements or accomplishments). Dowty [1979] provides a decompositional account of event structure in terms of the primitives BE, BECOME, CAUSE, and DO. He provides a model-theoretic interpretation for three of these primitives. His and further work has explored the syntactico-semantic consequences of these distinctions relating to aspect, postmodification, and diathesis alternation and refinements and extensions have been proposed (eg. [Moens and Steedman, 1988; Pustejovsky, 1995]). For example, as (1a,b) illustrates, durative PP postmodifiers can modify processes, but when used with transitions must be interpreted as modifying the resulting state. This can be captured in an event-based framework which posits subevent(ualities) of running and being at home broadly of the type proposed by Dowty, as sketched in (1c).

(1) a John ran for an hour.
  b John ran home for an hour (at lunchtime).
  c $\exists(e, e') \wedge run(e, j) \wedge become(e') \wedge at\_home(e', j) \wedge for(e', 1h)$
  d John ran (to the store)

Similarly, (1d) illustrates that a process can be coerced into a transition by the addition of a PP argument denoting a bounded path. Since the addition of a PP is often treated as a diathesis alternation, this shows that there can be a close connection between the phenomena of argument structure and event structure. Pustejovsky & Busa [1995] push this further by arguing that subevent headedness in transitions, that is whether the process or resultant state is the head, determines which subevent argument structure is realised syntactically. Many transition predicates are inherently marked for left/right headedness but when this is unspecified the resultant ambiguity over syntactic realisation leads to the causative/inchoative alternation. Pursuing the implications of contextual effects on aspectual interpretation has led some to argue that there are no inherent lexically-encoded distinctions and that the appropriate interpretation is entirely the result of the interaction of tense/aspect marking, choice of construction, modifiers and extralinguistic context (e.g. [Zaenen, 1993]). Nevertheless, some type of subevent decomposition still seems needed in order to account for scopal effects with modifiers, and so forth.

4.1.3 Qualia Structure

Pustejovsky [1991; 1995] attempts to integrate most work on lexical semantics into the framework of generative lexicon theory. However, the most innovative component of this theory, extending work by Moravsik [1975], is his proposal concerning qualia structure as an alternative to purely relational or decompositional descriptions of word senses. There are four qualia, *constitutive*, the relation between an object and its constituent parts, *formal*, that which distinguishes it within a larger domain, *telic*, its purpose or function, and *agentive*, factors involved in its origin. For instance, an entry for *book* might specify that it is constituted of text, its formal properties are that it consists of bound pages, a CD-ROM or whatever, its purpose is to be read and it was created by writing.

Qualia structure serves several roles in the theory. However, perhaps the main advantage of this approach to the specification of lexical meaning is that it provides a natural mechanism for co-specification of related polysemic senses. Thus, in examples like *he picked up and began to browse one of the books on the table* the verb *pick up* selects the formal physical dimension and *browse* the constitutive (or possibly telic) dimension. The fact that the two predicates are readily conjoined suggests that an approach which enumerates distinct polysemic senses of this type as separate lexical entries will be inadequate.

Qualia structure in nominals is also utilised in accounts of adjectival modification (*fast motorway*, *fast waltz*) and logical metonymy (*John began a book/cigarette*). Verbal qualia structure is utilised as a means of representing subevents in transitions; for example, the formal role of *break* is the resultant state of being broken, whilst the process of breaking which brings this about is treated as the agentive role. It is not clear to me that this extended use of qualia structure can have an identical semantics to the nominal cases, nor that it makes much sense to talk of the purpose or physical properties of eventualities in general. A more general question concerning the representation of word sense in terms of qualia is determining the semantic status of the various dimensions in a given utterance. Intuitively, it seems that one or more dimensions are asserted and the remainder presupposed in any given usage. However, this question has not been addresssed thoroughly to date.

4.1.4 Inheritance Structure

Inheritance structure is defined in terms of is-a or subtyping relations between elements of lexical entries. For example, the lexical entry for *novel* will inherit elements from *book* but will specialise the constitutive dimension from text to narrative. Inheritance allows for generalisation and economy in the statement of the lexicon as well as capturing relations between entries. The precise form of inheritance allowed is intimately connected to the LKRL adopted: [Pustejovsky and Boguraev, 1993] assume a fully general language such as KL-ONE in which multiple conflicting default inheritance is possible; [Pustejovsky, 1995] utilises subsumption in typed feature structures, thus limiting himself to multiple orthogonal (non-conflicting) non-default inheritance, as in HPSG ([Pollard and Sag, 1987; Carpenter, 1992]); [Copestake and Briscoe, 1995] motivate the use of multiple default inheritance in typed default feature structures for lexical semantic representation; [Evans and Gazdar, 1989b; Evans and Gazdar, 1989a] utilise multiple default inheritance of path equations yielding a system with similar expressive power, though reentrancy between paths in a feature structure cannot be directly manipulated.

In general, systems which make use of simpler inheritance schemes need another formal operation, usually dubbed a lexical rule, which generates derived lexical entries from basic ones. Lexical rules have been used to generate inflectional, derivational, diathesis alternated, and sense extended variants of basic entries. However, in most cases alternative analyses have been proposed which utilise underspecified representations and/or inheritance to generate the required variants. Pustejovsky & Boguraev argue that a system with the expressive power of KL-ONE allows for guided inference to lexically related concepts; for example, from the agentive dimension of *prisoner* 'imprison' it is possible to follow links to 'escape' and so forth generating a space of related and potentially relevant concepts. This step – from constrained inference in terms of is-a/default links between linguistically significant syntactic and semantic information, in the sense that it directly affects syntagmatic processes, to open-ended inference in terms of concepts – represents the boundary between linguistic and real-world knowledge and inference. To the extent that clearly linguistic lexical processes can be dealt with using a constrained framework such a distinction acquires more justification.

4.2 Lexical Semantics in Acquilex

The Acquilex projects have in part been concerned with the development of a computationally-tractable theory of the lexicon focussing on lexical semantics. In what follows, I summarise relevant publications and give pointers for further reading. The papers I reference are mostly available electronically at `http://www.cl.cam.ac.uk/Research/NL/acquilex/acqhome.html`.

4.2.1 Typed Default Feature Structures

The LKRL we utilise is an extension of the HPSG scheme to Typed Default Feature Structures (TDFSs). TDFSs are 'double' typed feature structures in which the second defeasible feature structure must be subsumed by the first indefeasible one (that is it must be strictly more specific in terms of the type

$$\begin{bmatrix} \mathbf{t}^1 \\ \mathrm{F} : \begin{bmatrix} \mathrm{G} : \mathbf{a}/\boxed{2} \\ /\mathrm{H} : \boxed{2} \end{bmatrix} \\ \mathrm{H} : \mathbf{b}/\mathbf{c} \end{bmatrix}$$

Figure 4.1: An Abbreviated Typed Default Feature Structure

system). An associative order-independent operation of typed default unification is defined over TDFSs. This operation can be used to implement default inheritance in a type hierarchy and also more generally allow any two TDFSs with compatible types to be unified. Specificity of defaults is defined in terms of the type hierarchy, so a default value associated with a more specific (sub) feature structure overrides the default value associated with the less specific one. This requires that TDFSs record the type of the (sub) feature structure which introduced a default value into a given TDFS in order to avoid ordering effects from a series of unifications. Nixon diamonds cannot arise in the default inheritance system based on the type hierarchy as only orthogonal inheritance is allowed. However, unification of two conflicting default values from TDFSs of the same type will result in the least specific default value (i.e. $\top$); that is, the system is sceptical.

The underlying logic and formal details of TDFSs and the associated operation of typed default unification is given in [Lascarides *et al.*, 1994] and in [Lascarides and Copestake, 1995]. Here we give some informal examples and motivation for these extensions to the formal framework proposed by [Carpenter, 1992]. TDFSs can be abbreviated by collapsing identical parts of the double feature structures and marking the distinct default components by a slash, as in Figure 4.2.1 in which the reentrancy between path F:G and H is default as is the path consisting of the attribute H. We ignore details in what follows since they are identical to the default component of the TDFS in the initial description.

Default inheritance in the TDFS system can be used to capture the fact that many lexical hyponymy relations are not absolute; for example, the telic role of a *book* is 'read' and *novel*, *monograph* and *dictionary* might all be defined as subtypes of *book*. However, in the case of *dictionary* the telic role will need to be overridden to 'refer-to'. Furthermore, as we shall see below there are reasons for thinking that the specification of telic roles as well as other aspects of lexical semantic specification must remain default 'beyond the lexicon'. The approach to defaults exemplified by the TDFS scheme, in contrast to most other approaches to default unification, allows exactly this by explicitly marking default specifications and ensuring that this marking persists through the default unification operation.

In addition to inheritance via subtyping, the TDFS lexical framework incorporates lexical rules which generate derived lexical entries from basic lexical entries represented as TDFSs. Most lexical rules are specified as type-to-type mappings without further constraints on the content of the input and output TDFSs beyond those specified in the relevant type declarations. In order to constrain the application of lexical rules to account for exceptions but allow for novel as well as conventionalised application (that is, deal with semi-productivity), a probabilistic interpretation is placed on TDFSs which means that lexical rules may generate 'neological' unattested entries with very low probabilities associated with them. Further details are provided in [Briscoe and Copestake, 1995] where the approach is exemplified with reference to verbal diathesis alternations.

4.2.2 Systematic Polysemy

Any approach to lexical semantics which wishes to break away from the inadequate and unrevealing sense enumeration paradigm ([Pustejovsky, 1995, 39f]) needs to distinguish systematic polysemy from idiosyncratic polysemy and homonymy. For example, the polysemy of *book* between the physical object and what it represents seems to be common to all words for representational artifacts not only including types of book but also films, pictures, and so forth. It is this type of lexical semantic ambiguity which a more adequate account of the lexicon should express generatively. [Copestake and Briscoe, 1995] and [Copestake, 1995] argue that systematic polysemy can be a result of constructional polysemy or semi-productive sense extension elaborating [Cruse, 1986] and others' informal distinction between sense modification and sense change.

Constructional Polysemy

Constructional polysemy is a consequence of sense modification in context where modification can be a consequence of further specification of an underspecified lexical entry, broadening of a specific entry with a default component, or coercion of quale to asserted status through semantic selection by predicates. (2a) represents a case of underspecification where the telic role of *reel* is '(un)wind' but the nature of the material to be (un)wound is unspecified and acquired from context.

(2) a The film/fishing/cotton reel broke.
    b The cloud (of flies) was dense.
    c John enjoyed the book.

(2b) represents a case of broadening where *cloud* has a default constituency of water vapour, but this can be overridden by an *of* complement. (2c) is a case of type coercion of *book* by the predicate *enjoy*, which selects for an eventive complement (and, by default, the one specified by the telic role – in this case 'reading'). The availability of both senses of *book* in the case of type coercion is illustrated by the possibility of what [Pustejovsky, 1995] terms co-predication – *John picked up and finished his beer* – supporting a treatment in which both senses are accessible from a single lexical entry. However, our current account does not deal with cases of coercion which involve non-constituent coordination, such as *John enjoys films and mending antique clocks* as a consequence of our decision to treat the coercion as internal to *enjoy* in order to deal with co-predication.

We argue that in these cases there is no question of semi-productivity or semi-regularity. Rather, if the appropriate linguistic context occurs, the sense is modified in entirely predictable ways. We provide a fully formal account of each of these processes in the TDFS framework which serve to flesh out and formalise a number of mechanisms and phenomena which [Pustejovsky, 1995] discusses under the rubrics of co-composition, co-predication and type coercion. The regularity of the polysemy in type coercion of course only extends as far as the eventive interpretation of the bare NP. The precise event is open to further contextual specification as in *Judging by the number of pages missing, my pet goat obviously enjoyed that book.* I return to this below.

Semi-Regular Sense Extension

Semi-regular sense extension can be distinguished from constructional polysemy on the basis that it results in generation of a variant lexical entry by lexical rule and such rules are not fully productive. For example, the various conventionalised subcases of nominal 'grinding' which yield mass nouns denoting a substance originating from the count noun sense, such as wood-grinding (*oak tree*, *oak table*), and food-grinding (*a haddock*, *haddock fillet* are quite regular within the appropriate lexical classes. However, there are exceptions caused by blocking (*cow/beef*, *pig/pork*), that is, pre-emption by synonymy. Furthermore, the general process of grinding, though applicable in the limit to any count noun denoting an individuated entity has very variable acceptability/conventionality, and conventionalised subcases differ cross-linguistically, suggesting the need for a linguistic overlaying of an ultimately more general conceptual transfer. An indication that grinding involves sense change rather than sense modification comes from the relative unacceptability of co-predication as compared to the book/contents polysemy discussed above: *?Sam fed and carved his lamb*, *Sam picked up and enjoyed that novel.*

Similar arguments can be made concerning the treatment of diathesis alternations: these also involve fairly regular sense extensions but are subject to lexical gaps which are not entirely predictable by restricting the application of lexical rules to semantic classes, even if these classes are narrowly specified (e.g. [Pinker, 1989]). [Briscoe and Copestake, 1995] show that verbs which undergo the dative alternation cannot be accurately specified in terms of narrow semantic classes, though a broad class specification is certainly necessary to restrict application of the rule. Nor does it seem likely that exceptions can be accounted for entirely in terms of blocking. Instead we propose that ultimately, acceptability is judged in terms of attested frequency and that low frequency lexical entries are utilised only in neologisms, for rhetorical effect or whatever.

The account of lexical rules in the TDFS framework captures semi-productivity by incorporating conditional probabilities that word forms will be associated with lexical entries into the control component of lexical rule application and providing a means for computing the overall probability of an interpretation. Probabilities are not a component of the TDFS LKRL but are more appropriately seen as part of the performance component of a language processing system. Thus, the acceptability of *He designed Bill a new house* as opposed to *He created Bill a new house* in American English is related simply to the non-occurrence of *create* in this construction in the experience of American English speakers and not to blocking or semantic differences between the two predicates. On the other hand neological use of *create* in this construction is interpretable because the lexical rule of dative can apply to *create* even though the resultant lexical entry is marked as unattested and of consequent low probability.

Blocking of sense extensions is not absolute but rather carries additional implicatures when it is overridden, as in *The drunken revellers were all too eager to tear into the roast cow.* In this sense blocking is a barrier to the establishment or institutionalisation of a sense rather than to its use. [Briscoe *et al.*, 1995] provide an account of blocking which captures the pragmatic markedness of unblocking. The probabilistic account to some extent subsumes this within a more computationally tractable framework, but fails to deal with the implicatures generated by unblocking.

4.2.3 Semantic/Pragmatic Interactions

The framework of lexical semantic description we have developed is intended to be compatible with extant approaches to (computational) compositional semantics. Lexical semantic representations are couched in an event-based version of a default conditional variant of the predicate calculus, which is itself described using TDFSs. [Lascarides, 1995] and [Lascarides *et al.*, 1996] following [Frank and Reyle, 1995] interpret these TDFS descriptions within an extended version of discourse representation theory integrated with an approach to discourse interpretation based on a conditional default logic (SDRT/DICE, [Lascarides and Asher, 1993]). Thus, the process of compositional semantic interpretation can be implemented using successive applications of typed default unification, and this allows us to formalise underspecification, broadening, coercion, co-composition and co-predication as an interaction between lexical semantic specification and compositional semantic interpretation.

The semantic part of a TDFS may contain default specifications which have persisted from the lexicon through compositional semantics. For example, *John enjoyed the book* will have a logical representation like $\exists(e, e', x, y) \wedge john(x) \wedge enjoy(e, x, e') \wedge\ book(y) \wedge\ P(e', x, y) \wedge\ *read(e', x, y)$ where P is a predicate variable over event(ualitie)s and **read* states that this is a reading event by default. For an example like *My goat enjoyed that book* this lexically-specified default is pragmatically inappropriate, so the interface to discourse interpretation must allow for such defaults to be overridden pragmatically. [Lascarides, 1995] gives a DRT-based semantics for the asterisk notation and shows how it can be integrated into the DICE framework in terms of two general principles: Defaults Survive and Discourse Wins. These two principles ensure that in the absence of conflicting discourse/pragmatic information lexically-specified defaults survive in the logical representation of the meaning of the input produced during parsing. However, in the presence of such conflicting information, for example that goats don't read, the discourse component both overrides the lexically-specified default and can supply a more appropriate refinement of the event variable in terms of more specific pragmatic information, for example, that goats eat indiscriminately. [Briscoe *et al.*, 1990] provide some empirical evidence in support of this account of the lexical semantic / discourse processing interface, by demonstrating that logical metonymy is utilised only a) where the lexically-specified default interpretation is correct but the context may be uninformative or b) where the local context clearly overrides the lexically-specified default. Otherwise a non-metonymic complement is utilised. Less modular accounts of this interface (e.g. [Hobbs *et al.*, 1992]) cannot explain the subset of a) cases in which the context is neutral other than by incorporating a lexical default into the pragmatic component. Evidence that these really are lexical defaults relating to word senses rather than concepts *per se* comes from examples like *?John enjoyed the doorstop* where although *doorstop* denotes a book, the telic role for the this lexical item does not provide a coherent refinement of the metonymic event leading to oddity.

The approach reported in [Lascarides *et al.*, 1996] extends this account of the lexical semantic / discourse processing interface to account for coherence effects in co-predication and to provide a more adequate account of zeugma. The absence of crossed readings in examples like *John has a file and so does Bill* and the zeugmatic effects in ones like *John banked the money and then the plane* are usually explained in terms of the 'inaccessibility' of multiple lexical entries (senses) in coordination constructions. However, examples like *This chicken is corn-fed and delicious* or *This thesis has thousands of pages and is unreadable* which are not zeugmatic but involve polysemic senses suggest that this account is incomplete at best. In our framework the related senses of *thesis* are a consequence of constructional polysemy and are represented within one entry, but the senses of *chicken* result from the lexical rule of meat-grinding, suggesting that the former but not the latter should be zeugmatic given our theory combined with the standard account. However, it is also possible to construct zeugmatic examples using the same senses of *thesis* and *chicken* – *This chicken is beheaded and delicious*, *This thesis is excellent and on fire* –

suggesting that the standard account is not capturing the relevant constraint. The causal relevance of the two conjuncts is critical to the discourse coherence of these coordinations. DICE captures these conditions in terms of conditions on the inference of the discourse relation *parallel* which is triggered by clausal coordination. It is shown that the interaction of polysemic senses with these conditions is what blocks crossed readings and causes zeugma.

4.3 Conclusions

The approach that we have developed to lexical semantics and the organisation of the lexicon generally contra the proposals of [Hudson, 1995] and of most computational systems (e.g. [Hobbs *et al.*, 1992]) is modular in that we argue for a restricted LKRL with well-defined and restricted interfaces to syntax, compositional semantics and pragmatics. This more modular account offers several theoretical and computational advantages, resulting in a stronger, more falsifiable theory which can be couched in a computationally-tractable formalism. Further research will be needed to decide whether the framework is empirically adequate. A number of immediate gaps and inadequacies can be identified: the interface between TDFS and DICE has not been implemented, the account of (un)blocking requires further work, and some types of non-constituent coordination involving type coercion remain outside the capabilities of the current proposals.

Chapter 5

Computational Semantics: Significant Themes and Future Directions

5.1 Introduction

The aim of the FraCaS project has been the exploration and realization of a framework for computational semantics. Results of our attempt to reach this goal are, among other things:

- a presentation of different available semantic theories, and a synoptic analysis of their appropriateness to cover different classes of phenomena (D8 and D9),
- the specification of a common logical framework, as well as the development of a set of conceptual tools for the purpose of computational semantics (D15),
- the realization of an educational and research tool which incorporates different grammatical and semantic formalisms, and different methods for meaning composition, as well as a test suite for the inferential behaviour of systems in computational semantics (D16).

In addition to these concrete deliverables, the FraCaS project has led to a better general understanding of the field. To be sure, there still are important problem areas in computational semantics which are in need of intensive further investigation. In the first part of this paper, we give short characterizations of what we take to be the current main issues in computational semantics.

Also, we will outline our view about future work to be done in computational semantics. There are concrete research tasks which immediately result from FraCaS work, but there also is a quite concrete picture of mid-term research that should be taken up after FraCaS. We will comment on these two kinds of follow-up tasks in the final part of this document.

Before looking into open themes and future research tasks, however, we want to point out in short what in our opinion have been the most important insights of the project work which directly resulted from bringing together researchers in semantics from different traditions and persuasions. The following information was glossed from several personal contributions on what the individual FraCaS participants thought were the main insights and difficulties in current research in computational semantics.

5.1.1 The Importance of Notation

Logicians tend to ignore or downplay issues of notation, as for them the logical representation language only is a pointer at modeltheoretic reality anyway. During the FraCaS project we have found that in computational semantics, notational issues are important, for at least the following reasons (it should be obvious, though, that these reasons have very different status):

Computation The point should be stressed that semantic representations are the object of semantic *processing*, and that different notations might have different effect on the efficiency of processing. It is often a non-trivial matter to specify a translation mapping from surface syntax to some medium of logical form. This is almost certainly the case if one is developing an extended fragment. In such cases, changes of notation should be guided by computational considerations having to do with the design of the syntax-semantics interface.

Cognitive Considerations One kind of notation may fit assumptions about what the cognitive process of incorporating new information into an already existing body of knowledge than another kind.

Familiarity The notation of a formal representation language should be such that working linguists can feel at home with them. It turns out that a shift in notation may make the working linguist ill at ease.

Tradition Different schools in formal semantics adhere to different notational conventions, and in many cases, people working in a particular tradition do not fully grasp the reasons behind certain notational choices. It is understandable in such cases that they stick to what they are familiar with.

5.1.2 Semantic Parallels

Here is a short list of mechanisms which have similar functions in the various theories.

Dynamic Reference to Individuals The difference between the role of reference markers in DRT with the way in which variables function in classical logic and in programming languages has become much clearer. Use of discourse referents in DRT seems related to the use of mechanisms for simultaneous abstraction and parameters (defined in such a way that alpha conversion is lost) for dealing with anaphora. It is time now to explore dynamic typed logics (or: lambda calculi with explicit substitution and without alpha conversion) and connect them to various proposals for compositional dynamic semantics (the various combinations of DRT or DPL with lambda abstraction).

Contexts Parallels have been drawn between the contexts in dynamic semantics, abstracts in STDRT, and DRSs.

Variants on Composition The relationship has been clarified between different variants of functional composition: e.g. generalized functional application in Lambda DRT, Minimal Indexing Assignment in Situation Semantics, Composition as used in Categorial Grammar.

Dynamic Information Processing The need for discourse based information as well as purely semantic information for dealing with anaphora is something which all the theories stress.

5.1.3 Issues to be Addressed in a Theory Independent Way

One insight of FraCaS was that a toolbox of semantic techniques can use tools from different boxes, in the sense that specific techniques developed inside a particular school can often survive in a quite different context. Here are some examples:

Resource Situations Developed and inspired by situation theory, but a useful tool in its own right. Does not depend on 'deep' features of situation theory. The resource situation example illustrates that frameworks often are a kind of nursery ground for solutions that can also survive outside of the conceptual environment in which they had their early development.

Information Updating Perspective on natural language processing that is very much part and parcel of dynamic semantics and DRT, but the main insight that pieces of discourse often are processed in the light of what has been processed before can be incorporated in any system of natural language semantics.

Semantic Composition As recent work on compositional versions of DRT show, many aspects of the Syntax-Semantics Interface that were at some point in the past thought to be essentially non-compositional, can be rephrased in a compositional way.

Underspecification Turned out to be a topic amenable to handling in different but converging ways from different theoretical perspectives. In particular during the second year of the FraCaS project work, a significant effort was made to make headway in this area. It turned out to be possible to address issues such as representation of different forms of underspecified information, processing underspecified representations (disambiguation in context) and reasoning with underspecified representations in a theory-independent way.

Paradoxes and Propositionhood The danger of paradox acts as a counterbalance to the natural urge to propose more and more expressive semantic frameworks. No paradigm in natural language semantics is immune to this danger, and the efforts to stave it off point towards a natural definition of propositionhood (probably worked out most fully right now in property theory).

The FraCaS collaboration was not aimed at substituting a new theory as a kind of grand synthesis for existing theories of semantics. It has even encouraged the participants to stick to their chosen approaches or 'schools'. It was very much felt that the various semantic schools of thought are the greenhouses where solutions to semantic problems can grow until they are strong enough to be transplanted to (reformulated in terms of) a different paradigm. This insight explains the tendency of researchers in semantics to stick to their chosen paradigm as understandable, rational and useful. We will now turn to the significant themes in computational semantics, as we view them, that have emerged from the FraCaS collaboration.

5.2 Significant Themes in Computational Semantics

5.2.1 Natural Language Semantics and Inference

There can be no semantics without logic. This is as true for the semantics of natural language as it is for that of artificial languages. But the point merits special stressing when it comes to natural language semantics, since there it has a tendency to be forgotten.

The intrinsic connection between natural language semantics and logic is easy to see when one assumes that it is a central task for a semantic account of natural language to describe how the surface forms of linguistic expressions can be transformed into semantic representations. The point of this transformation is to obtain representations that are "logically transparent", representations which are accessible to well-understood systems of logical deduction by means of which one can derive their logical consequences; for it is precisely in this respect that semantic representations are meant to distinguish themselves from, for instance, "purely syntactic" representations. Computationally, the point of a semantic representation is that it enables a theorem proving system to compute its consequences from it, or to verify whether a putative conclusion does follow from it or not.

It is important to realize that the view held by many linguists and philosophers of language - that the principal role of semantic representations is to obtain logically transparent accounts of conditions of reference and truth - does not affect the essence of this conclusion. For the empirical bite of a theory of semantic representation thus conceived lies in the possibility of verifying its predictions; in verifying, that is, whether e.g. the truthconditions which the semantic representation assigns to a sentence s match speakers' intuitions about the conditions under which s is true. Computationally this amounts to the possibility of verifying, for any logically transparent description of a situation about which s can be used to make a claim, whether the semantic representation of s logically follows from the situation description just in case s is judged to be true of the kind of situation described. Once more the point of the semantic representation is thus the possibility of exploiting it deductively - only that here it is a matter of what the representation follows from rather than what follows from it.

A similar conclusion can also be drawn concerning semantic theories which present themselves as assigning natural language expressions (or their syntactic analyses) conditions of truth and reference directly, without the intermediacy of "semantic representations". For again, such a theory has empirical content only to the extent that it is possible to check whether the conditions it assigns to particular expressions match speakers' intuitions. And in last analysis this is a matter of being able to verify whether, say, the truthconditions which the theory assigns to s follow from particular logically transparent situation descriptions. That the truthconditions which the theory delivers for s are couched in some general logical metalanguage (the language in which the semantic theory is formulated) shifts the logical import of the semantic theory from one formalism to another, but the import is there in the one case as much as in the other.

There are a couple of morals for computational semantics that can be drawn from this general assessment. First, there is little or no point in developing a theory of semantic representation unless one is prepared to develop a theory of inference to go with it. This does not mean that it is pointless to develop an account of semantic representation (or for that matter, an account of truth and reference conditions) before an accompanying account of inference from and to semantic representations (or metalinguistically formulated truthconditions) is in place. But at the very least the development of a matching proof theory must be envisaged as part of the theory as a whole, and the theory will have to be considered incomplete until its proof-theoretic component is in place too.

Second, by much the same token, there is no point in developing NLP systems which take the trouble of constructing semantic representations, if they have no inferential components which can make use of those representations. This should not be taken as a plea for NLP systems which stay clear of semantics as well as logic. In fact, we consider it most unlikely that sophisticated language understanding systems, such as high quality machine translation systems or automated question answering systems, could be designed in which logic and semantics were to play no significant role. What we mean to stress is that semantics without logic is otiose. Similarly, fast construction of semantic representations would be of limited value, in case the complementing theorem provers are slow and inefficient.

What ought to be done? Develop formalisms for semantic representation and algorithms for representation construction for as wide a repertoire of words and syntactic constructions as possible, but never without losing sight of the complementary task of developing the corresponding inference engines.

5.2.2 Inference for Constructing Semantic Representations

In the previous section we have emphasized the inseparability of semantics and logic. As a matter of fact, recent experience with problems in natural language semantics, especially in the domain of the interpretation of discourse, have made clear that semantic representation and inference are connected in an additional way which we haven't mentioned yet: The construction of semantic representations of discourse itself requires, at almost every turn, the use of inference.

One way to explain the need for deduction in representation construction is to point to the well-known ambiguity problem. When we compute the "potential ambiguity" of a phrase by multiplying the ambiguity degrees of all the lexical items it contains with the ambiguities inherent in the syntactic constructions by means of which the phrase is put together the numbers one arrives at are often surprisingly high, even for phrases of moderate size. (For expressions of less than twenty words they may run into the thousands.) Nevertheless, in ordinary language use, expressions of such or even larger size rarely seem to present the human user with serious ambiguity problems. While it does happen that people misinterpret what they hear or read, or, alternatively, that they do not know what to make of it because there is an ambiguity they are unable to resolve, this is the exception rather than the rule. For the most part, they succeed in reducing the dizzying multiplicities of meaning down to the single interpretation which the user of the expression had in mind.

A cursory reflection on how this might be possible - and closer investigation of the phenomena confirms this - suggests that in many and perhaps all cases disambiguation results through elimination of all the unintended interpretations by showing that they are inconsistent or implausible (either phrase-internally or relative to the context in which the phrase is being used).

Algorithms for constructing semantic representations of discourse must have the same powers of disambiguation as human interpreters and thus they too must be able to carry out logical inferences at almost every step of their operation. Moreover, a more careful look at particular cases where inferences are needed in representation construction has shown that they have to de drawn on the basis of incomplete or underspecified representations. (See in this connection also the section on underspecification below.) Thus the inference devices that the construction algorithm can call on should be able to use as inputs, and sometimes return, such underspecified or incomplete representations. In addition, there are growing indications that the inferential support of construction algorithms should not come in the form of a single all-purpose inference machine, but rather of a variety of specialized inference engines, which are designed to deal only with inferences of specific kinds but which deal with those kinds of inference efficiently. Finally, it must be kept in mind that many of the inferences needed for disambiguation are default inferences rather than logically hard deductions.

What is to be done? Develop the monotonic and non-monotonic inference engines needed to support the construction of semantic representations and, as a preliminary to this, the special and partial logics and proof systems of which these inference engines are the implementations.

5.2.3 'Non-logical' Reasoning

It is often claimed that the role formal logic plays in the semantic processing of natural language is a modest one at best. For most of the information that is typically conveyed by means of natural language appears not to be deductive in a strict and straightforward sense: rather, inferences that people draw while or after interpreting language are typically of a probabilistic or inductive variety; or they derive in part from defeasible assumptions; or are obtained by means of defeasible principles.

Superficially this assessment of the place of formal logic within a semantics for natural language seems to contrast starkly with the central importance we have been attributing to it in the two preceding sections. Yet, how much of a contrast there really is depends on precisely what we take the nature and scope of formal logic to be. On a narrow, "classical", conception, according to which logic is identified with the syntax, semantics and proof theory of the predicate calculus, the position presented in the preceding sections would clearly fly in the face of the sceptical view expressed in the paragraph above. But it should have been clear that this is not the conception of logic which underlies that position. For one thing, we mentioned, in 5.2.2, the importance of "non-monotonic" or "defeasible" inferences and of logics which deal with them; for another, we emphasized the need for a range of special purpose inference engines, each implementing its own "logic". Both pleas reflect the much more catholic conception of logic that has increasingly become the norm over the past two or three decades within the communities of philosophical logic and artificial intelligence - a conception in which there is room for an unbounded variety of logical systems, differing from each other in their expressive power, in their (semantic) characterization of valid inference or in the (proof-theoretical) means which they offer to capture or approximate those characterizations of validity.

In fact, natural language semantics has made its own contributions to the ever more diversified "landscape" of logical systems that presents itself today to the bewildered eye of the uninitiated. Its most important impulse so far has been for the development of representational frameworks (or "logical languages") that are capable of correctly representing information that is expressed in natural language, or of representing such information in a form that allows for systematic translation from their natural language expression into the formalism.

In fact, at least two of the approaches towards natural language semantics that FraCaS has paid close attention to, Situation Semantics and Discourse Representation Theory, involve the use of logical languages which deviate from the standard of predicate logic in just this way. Moreover, as a comparison between the two shows, such approaches can lead to distinct characterizations of logical consequence as well (classical Situation Semantics proposes a weaker-than-classical logic in which e.g. the law of excluded middle does not hold), but they do not need to (standard DRT has adopted the logical consequence relation of classical predicate logic, even though it represents predicate-logical content in a somewhat different form).

Recent and continuing research on the logical representation of various aspects of natural language meaning (again, some of it within the context of FraCaS) has added and keeps adding new items to the repertoire of alternative "logics". Special mention should be made in this connection of current work on the logically transparent representation of intentionality, i.e. of the vast spectrum of different ways in which linguistic content can be present to the human mind (both the mind of the language user and the minds of those about whom he speaks). Not only classical predicate logic, but also the by now familiar systems of intensional logic (such as Montague's Higher Order Intensional Logic), whose design was prompted by this very problem, do not deal with it adequately; and the development of logical alternatives that do a better job of representing intentional contexts continues to be one of the most important challenges that natural language semantics has to meet.

Still, intentionality is only one of those challenges. Another one, often cited as a difficulty that formal logics are constitutionally unable to cope with, is vagueness. One of the fundamental tenets of logic, so the argument goes, is that whatever is worth saying can be said exactly, and this leaves the bulk of what is transacted in natural language by the wayside. As a claim about the classical systems which have dominated formal logic since the time of Frege this contention is pretty much accurate. Those systems were developed by mathematicians for the sake of formalizing mathematics, in a conscious effort to eliminate all scope for unclarity from the outset, so as to force the user to free himself while formalizing a part of mathematics from all the muddles and confusions which to which we are prone when we operate within an informal setting. But much has changed since the days when it was common, and largely right, to identify formal logic with the predicate calculus. While it seems fair to say that the problems of representing and reasoning with vague concepts have not yet found a definitive, generally accepted solution, the existing results in this area clearly refute the argument that the ubiquitous presence of vagueness in natural language should make it inaccessible to analysis with the tools of formal logic. (The demonstration that for many purposes it is possible to reason Witt vague concepts as if they were sharp - that is, that, within certain limits, the logic of vague concepts is the same as that of sharp concepts - may serve as a simple but relevant example.)

Another important feature of natural language which sets it apart from the architecture of classical logic, is its context-dependence. The semantics of countless natural language expressions, simple as well as complex, involves context sensitivity of one kind or another. This too is a topic in natural language of which we have not yet by any means seen the end; but it is also an aspect of the semantics and logic of natural language that has been the topic of formal investigations for many years - from the work of the sixties and seventies on tense and modal logic to recent work involving, among other themes, the semantic import of temporal, spatial and personal perspective and “quantification over contexts”. The latter end of this spectrum is directly connected with a feature of natural language semantics that has been especially prominent over the past years, and which has played a major part also within the activities of FraCaS, viz. the discourse connections (anaphoric and other) whose study has led to the current theoretical paradigm of Dynamic Semantics. From a logical perspective the dynamic dimension to natural language meaning can be treated in a number of different ways, each giving rise to its own consequence relation- compare for instance, the update logic of Veltman or the arrow logics of Van Benthem with the purely classical logic of DRT. Here too, we see how studying the mechanisms by which information is encoded and transmitted in natural language has led to an enrichment of our understanding of the spectrum of logical options and to a broadening of our conception of what logic is - in stead of confirming the sceptical view that context-sensitivity is a possibly impassable obstacle to the application of formal logic to natural language.

While the need to express hitherto inaccessible dimensions of meaning has pushed us in the direction of more powerful logical frameworks, the need to control the problem of inferential complexity - How time or space-consuming is it to find or verify the inferences that language processing systems need to draw? - has pulled us in the opposite direction - the direction of fragments of existing formalisms, which are large enough to express what must be expressible for the particular purposes at hand and yet constrained enough to render inferencing tractable.

This search for manageable fragments of existing logical frameworks - or, alternatively, for genuinely new modes of representation of a kind that promises inferential tractability, but which one may then discover to correspond to subsystems of existing logics - has, in parallel with the drive towards an increasingly comprehensive repertoire of representational possibilities, bequeathed us with a growing thicket of logical systems. One of the projects to which FraCaS has addressed itself (see in particular Chapter I.2 of Deliverable D15) is the comparison and classification of these different logical systems. This comparative and classificatory effort requires a logic of its own - a comprehensive "metalogic", strong, transparent and uncontroversial enough to be acceptable as a vehicle of analysis which does not load the dice in advance, by excluding some systems or by unfairly favouring others. Thus we find, at the meta-logical level, a need which echos the concern of the great logicians from the end of the last and the first half of this century - the need for a single characteristic universalis which exacts precision from the user while allowing him to analyze all existing modes of representing meaning, and, so is capable of expressing in a clear and unambiguous way, at least indirectly all that can be expressed in any one of these modes.

There is a further feature of reasoning with natural language which has been seen as seriously limiting the usefulness of formal logic in modelling such reasoning. We touched upon this in the opening paragraph of the present section but did not expand on it yet. As a rule the inferences that are drawn from natural language premises depend not just on those premises alone but also on information about the meaning of individual words and on so-called "world knowledge". Realistic models of natural language inferencing must therefore incorporate large databases, data bases which it has proved to be extremely difficult to compile and structure in a way that facilitates search for premises relevant to particular inferential tasks. An additional complication is that much of the world knowledge that is needed to support such inferences is defeasible: conflicting information can override it, but so long as no such information is explicitly available we may rely on it. (It is primarily because of this defeasible character of so much world knowledge that reasoning with natural language is on the whole non-monotonic.)

The apparently Herculean dimensions of the task of building the requisite base of lexical and extralinguistic knowledge has been given as yet another argument for the subordinate role which logic is supposed to play in the interpretation and informational exploitation of natural language. Even if logic does play its part when sentence or discourse content is to be combined with the relevant lexical and extra-linguistic premises, its part is that of the very much lesser part (indeed, the very much lesser part) of a problem that is of huge perhaps intractable complexity over-all. (And on top of that, the logic we need is not some well-established deductive logic, but some kind of non-monotonic logic.)

For those who believe in the importance of logic for computational semantics there is only one answer to this objection: Build the large knowledge bases and investigate and model the strategies for defeasible use of those knowledge bases. As is well-known to all who have tried their hand at developing natural language processing systems which are capable of non-trivial semantic and logical processing, this is probably the hardest, and certainly the most time-consuming part of building such systems, and the principal reason why nothing truly satisfactory of the kind as yet exists. Still, it is a task that will have to get and keep our full attention.

What is to be done?

- Develop formalisms that can represent aspects of NL meaning that are beyond the representational capacities of formalisms currently in existence.
- Develop special purpose logics for specific inferential tasks and design efficient inference engines implementing those logics.
- Develop a systematic, easily accessible classification and comparison of existing logics and, if necessary, adapt the meta-logic which this classification and comparison require.

5.2.4 Lexical Semantics, Default Reasoning and World Knowledge

A comparatively small part of this task, though it is one that is daunting enough in itself, is the development of a lexicon capable of supporting semantic representation and inference in ways which reflect the use that is made of lexical information by us. Developing such a lexicon is important also in that it promises insight into what is generally considered an intrinsic part of language itself, and not some kind of extra-linguistic knowledge which interacts with linguistic knowledge only at language's pragmatic fringes. In the present section we look a little more closely at the problems of lexically supported interpretation and reasoning.

In applications of logic to real world inferences we usually assume that the premises of an inference and its conclusion are phrased in terms of the same concepts. The archetypical inference concludes from *All men are mortal* and *Socrates is a man* that *Socrates is mortal.* This is not a realistic situation. In real life, we are given information like *Everyone of us kicks the bucket sooner or later*, and *Socrates is like any of us*, and we would like to ascertain that the conclusion *Some day, Socrates will be dead and buried* is warranted.

To reduce this inference to the simple syllogism that we started out with, quite a bit of preliminary processing is needed. The reference of *us* needs to be fixed. The idiom *to kick the bucket* needs to be related to *die.* The meaning of *sooner or later* needs to be fixed as relating the point of speech to some point in the near or distant future. It should somehow be derived from the second premise that it describes a situation where *Socrates* is alive. The conclusion should be related to the reference time of the second premise, and to process the contents of the conclusion, the relation between *being dead* and *being buried* needs to be established.

Or, to take a simpler example, how do we conclude from *Everyone danced* that *Everyone moved*? Again, for this we need lexical semantic information about the relation between *dancing* and *moving*, to the effect that events of dancing are invariably events of moving (or, to phrase it in the terminology of Situation Semantics, that dancing *involves* moving). This information should be glossed somehow from the lexicon. In this case, note that we need the additional information that *danced* occurs in a positive position in *Everyone danced*, in the sense that from *Everyone A-ed* and *A-ing involves B-ing* it can be concluded that *Everyone B-ed.* Compare the non-inference from *No-one danced* to *No-one moved*: this has to do with the fact the position of *A-ed* in *No-one A-ed* is a negative position.

It should be clear that the process of drawing realistic inferences from natural language texts involves knowledge about the logical relationships among concepts. These relationships will often have to be phrased in terms of default inference ('*Normally, being dead involves being buried*'), and real world knowledge (one should know, e.g., that Socrates lived in ancient Greece and that the ancient Greeks did have the custom of burial and would never leave dead bodies to the vultures).

Simple as these examples are, they should make clear that a clear line between lexical and extra-linguistic knowledge is difficult to draw. For instance, how do we decide if it is lexical or extra-lexical information that dancing involves moving? Part of the problem is that words of often lexicalize information that is inseparably connected with how our world happens to function. For instance, that an automobile has wheels (and is not, say, mounted on runners, like a sled) is connected with the contexts in which automobiles are used (on roads rather than on snow or ice. Yet, something that was mounted on runners but was in all other respects like a car we probably would not call an automobile, simply because it fails to satisfy a salient criterion for the concept. And inasmuch as that is so, having wheels must be seen as part of the meaning of automobile as most ordinary people would understand the word.

Thus our earlier judgement that lexical knowledge constitutes the lesser part of the total knowledge base that is needed to support reasoning with natural language may well require correction. For it may prove difficult, and perhaps impossible, to specify the lexical contents of words in a form adequate to support reasoning without taking on board a very large part of what we would rather see as encyclopedic knowledge as well.

5.2.5 Computational Semantics and Cognitive Faithfulness

On the one hand, computational semantics aims at providing tools for natural-language engineering. As a subject in the field of cognitive science, computational semantics also aims at providing insights into the organization of human cognition, i.e., into the way humans represent and process meaning information. The goals are different, and imply different preferences with respect to subjects and methods of research. However, the results may be mutually relevant. In particular, in the light of the fact that the efficiency and ease of human meaning processing diverges drastically from what is the state of the art in computational semantics, linguistic engineering might learn from empirical, psycho-linguistic research.

Unfortunately, there is a wide gap between the community of computational semanticists using logic-based formalisms on the one side, and empirical researchers dealing with human meaning processing, on the other side. Normal semanticists usually would not consider the meaning-related work based on empirical methods to be semantics at all. On the other hand, psycholinguists would usually *deny* that the kind of semantic representations, composition techniques and inference methods used in logical semantics of natural languages have any cognitive significance. The disconnectedness between the two fields of research becomes particularly striking if we compare the situation in semantics with the fruitful interaction between theoretical, computational, and empirical research which has been reached in syntax.

One basic problem on the semantic level seems to be the different methodological backgrounds of the disciplines, which lead to a complementary selection of subjects of the investigation: logic-based formalisms favor the description of those parts of meaning phenomena that are directly reflected in truth-conditions. They allow modelling of other important aspects of meaning, as the move to dynamic semantics based on the notions of information state and context change shows. However, they do not support the modelling of non-propositional, analogous aspects of meaning. On the other hand, empirical access to meaning phenomena is difficult in general, feasible to some degree for content words and referential expressions, but very difficult where complex meaning structure is involved.

There are attempts to provide frameworks that consider complex meaning structure and requirements of cognitive adequacy at the same time (e.g., "Cognitive Grammar", "Cognitive Semantics", or the theory of "Mental Models"). These approaches are no doubt interesting and stimulating for the task of developing a semantic framework with broad coverage. Their problem, however, is their methodological pay-off: they have to give up, or at least weaken, the rigid standards of both logic-based and empirical research.

As far as we can see, there is no framework in sight which is in principle able to cover all aspects of natural language meaning on a sound formal and empirical basis. However, several things can and must be done to narrow down the gap between formal and empirical research, and at the same time improve the practical usefulness of computational semantics. Some of them can be subsumed under the themes of "Lexical Semantics", "Inference", and "Non-logical Reasoning", and are addressed in the corresponding sections.

We would like to point at another important aspect, which concerns the modelling of the semantic composition process. There, the semantic theories which are available severely restrict the possible processing techniques, and they do it in a way that a wide range of modes of semantic analysis are excluded, which have some cognitive and computational plausibility especially in connection with the task of processing spoken-language "ill-formed" input" (see 5.2.7). There seems to be a need for decoupling the process of semantic analysis from the constraints imposed by the semantic representation formalism, and thus allow to formulate alternative processing methods. One promising way towards this aim is the consistent separation between a representation language and a description language level. This separation has been proposed under the heading of "glue language" in [Dalrymple *et al.*, 1993] and has also been considered in FraCaS in connection with the notion of underspecification.

5.2.6 Underspecification

Natural language can be conceived of as a system of rules that assigns an utterance an amount of semantic information on the basis of the linguistic information which can be read off the utterance, plus situational or contextual information of different kinds. An utterance is semantically underspecified if the given linguistic and contextual information is not sufficient to unambiguously determine the kind of semantic information which it was intended to convey. Formalisms for underspecified semantic representation offer techniques for the compact description of the possible range of representations, as an alternative to a listing or multiple disjunction of specific readings.

The development and investigation of underspecified representation techniques is highly important for Natural Language Processing tasks in two respects. Underspecification allows us to model the process of utterance interpretation in a sensible way, in terms of continuously adding constraints and thus keeping track of the decreasing range of possible readings, without ever having to spell the set of possible readings out (which may be very large or at certain stages even infinite). Thus, the successive exploitation of different knowledge sources for semantic analysis can be described, and incremental semantic analysis can be modeled in the way that pieces of semantically relevant information become effective as soon as they are available.

Underspecified representations may allow processing the result of a semantic analysis in the case that it is ambiguous, without processing each of its readings separately. A special application which makes use of this advantage is machine translation: in the case of scoping, underspecification in the source language often transfer to a corresponding underspecification in the target language. Thus, an appropriate result may be achieved without disambiguating. Of course, the most general and important use of semantic representations is in terms of inferencing. Thus, one of the most important questions in the field of underspecification is, to what degree direct deduction on underspecified representations is possible.

What is the state of the art, and what are the expectations for future development? There is no uniform answer to this question, since there are structurally very different kinds of underspecification phenomena.

The structurally simplest type are cases of local underspecification in an otherwise uniquely given global semantic structure, as in the case of lexical ambiguities, referential ambiguities, and "relational vagueness". This kind of underspecification is well-investigated: representation is straightforward (in terms of meta-variables standing for the ambiguous predicate), and composition of representations is unproblematic as well. A denotational interpretation, and, accordingly, the specification of a direct deduction concept, is not completely straightforward because there may be covariation among different pieces of a complex representation, but can be provided,e.g., in terms of a super-valuation semantics.

For a second type of underspecification phenomena, it is the global structure of the semantic representation which is left undetermined. The most prominent sub-case, scope ambiguity, has been the subject of intensive discussion through the last years. There are a variety of representation formalisms available. Proposals for a compositional construction of underspecified scope representations have recently been worked out outside and within the FraCaS project. Furthermore, there have been attempts to provide a compositional denotational interpretation.

Some of the techniques for this are intuitively not really adequate, because they identify ambiguity with object-level disjunction (see e.g. [Reyle, 1993]). Other are not really complete, because they make essential use of syntactic substitution in the interpretation process. Still, we think a direct denotational semantics for scope underspecification is in the cards, and recent FraCaS work is pointing the way towards sound and complete inference systems for representations with scope ambiguities.

There are additional underspecification phenomena, however, which cannot be subsumed under one of the two kinds of underspecification described above: semantic underspecification induced by ambiguous, incoherent, or incomplete grammatical information. Examples range from syntactic ambiguities (e.g., modifier attachment), to syntactically incoherent utterances, incompletely uttered or incompletely understood sentences in spontaneously spoken discourse. These additional cases of underspecification have been widely neglected, though some first steps towards a representation have been made in the FraCaS project. Semantic processing of this kind of phenomena will be extremely important in the context of spoken language systems.

5.2.7 Spontaneous Speech and Other Forms of 'Extra-Grammatical' Input

Spoken conversations are perhaps the primary use of natural language by humans, and therefore should tell us something very fundamental about the way humans process language. Yet, looking at the transcript of one of these conversations we cannot fail to notice that the language is 'ill-formed' according to traditional notions of grammar. Consider for example the following fragment, from the TRAINS corpus of transcripts collected at the University of Rochester:

(5.1)
```
15.1 M: so
15.2  : why don't you j/
15.3  : you immediately send
15.4  : engine uh / an engine right now
15.5  : from Elmira
15.6  : to Corning
16.1 S: okay
16.2  : so we'll say
16.3  : engine E2
17.1 M: okay
18.1 S: go to Corning
19.1 M: all right
19.2  : and then we need to get
19.3  : either boxcar from Dansville or from Bath
19.4  : which one's closer t / to Avon
20.1 S: uhm
20.2  : to Avon
```

```
        20.3  : Dansville is closer
```

Utterances in spoken conversations are largely fragments, rather than full sentences; these fragments are mixed with pauses and other hesitations, with repetitions, and corrections of what has just been said. An utterance is often the result of a collaboration among the speakers, as in 16.1-19.4. The participants to the conversation are apparently able to successfully extract an interpretation from these fragments and editing signals; in fact, they do not seem to realize how much work they have to do. This form of language use is a challenge for theories of language processing and for NLP systems at all levels, from speech recognition to semantic interpretation; and it is most specially a challenge for theories of semantic interpretation that depend on syntax to guide the process of 'meaning assembly'.

So far, three main approaches to the problem of interpreting spontaneous speech have been adopted in implemented spoken dialogue systems. One approach involves developing a 'grammar of errors': interpretation proceeds as usual except that the system's grammar includes rules for dealing with repairs and hesitations in addition to the 'normal' rules. For example, utterance 19.4 would be processed by a rule that says that a PP can consist of a false start, an hesitation, a preposition and an NP. The problem with this approach is that it makes the traditional problems of parsing—ambiguity and hand-coding rules—even bigger than they usually are.

A second approach involves an 'editing' step between speech recognition and syntactic / semantic interpretation; the result of this editing step is a sentence without repetitions and repairs, that can then be processed by the rest of the system using standard techniques. One problem with this approach is that often enough the correction part of a repair makes reference to information contained in the 'repaired' bit. An example is the *it* in utterance 13.5 of (5.2), that refers to *the engine at Avon* which is part of *we should move engine E1 to Bath* which is replaced.

```
(5.2)   9.1  M: so we should
        9.2  : move the engine
        9.3  : at Avon
        9.4   : engine E
        9.5   : to
        10.1  S: engine E1
        11.1  M: E1
        12.1  S: okay
        13.1  M: engine E1
        13.2  : to Bath
        13.3  : to /
        13.4   : or
        13.5  : we could actually move it to Dansville to pick up
```

```
the boxcar there
```

Finally, semantically driven grammars have been used. Such grammars rely on expectations about the domain, in the sense that they use the semantic type of fragments to determine their role within the semantic interpretation of the utterance. Systems working this way tend to make no use at all of 'syntactic' information. These systems have some success in narrow domains, but it is an open question whether they can be applied to cases of input with a complex semantic structure e.g., input containing scope bearing elements such as negation.

In order for semantic theory to be applied in more robust systems for spoken dialogue processing, a theory of interpretation has to be developed which is consistent with what we know about the semantic properties of natural language expressions, yet does not entirely depend on the availability of a syntactic analysis, and is capable of taking into account the role of expectations in processing ill-formed input. Some systems (e.g., the TRAINS-95 system) already use both a 'traditional' and a 'semantic' parser working in parallel and use heuristics to choose among them; the goal is to achieve a smoother integration of these two kinds of techniques. It would appear that an essential ingredient of theories dealing with partial input will be a theory of underspecification, i.e., a theory of partially disambiguated interpretations (see 5.2.6 above).

5.2.8 Statistical Semantics

Members of the speech community sometimes express scepticism about rule based processing to derive meaning representations for sentences. They object that such an approach shares (at least) the problems of rule-based syntactic processing: it is fragile, knowledge intensive, and narrow in coverage. This view is best crystallised in the remarks often attributed to (now former) members of the IBM speech processing group, namely, that all those linguists and computational linguists currently engaged in building rule-based systems could be more profitably directed towards the semantic annotation of large scale corpora. The implication is that it is only lack of sufficient training data that prevents the statistical methods applied by this group to speech recognition, tagging, parsing, and translation, from being applied with equal success to the task of assigning meaning representations to sentences.

One can question some of the assumptions that lie behind this attitude. For example, it is by no means clear that rule based systems for syntactic processing can never approach the accuracy and robustness of statistical systems. Several groups have reported results from rule based systems for tagging or phrasal analysis that are as good or better than those attained by statistically driven systems ([Brill, 1994], [Karlsson, 1995]). Also, the distinction between rule based and statistical systems becomes harder to draw as more sophisticated kinds of analysis are considered. For example, the Core Language Engine uses an explicit grammar to assign parses (and meaning representations) to sentences and then uses statistical preference measures to choose between alternative analyses. A parsing system like that reported by Black (see [Black, 1993]) uses a grammar derived automatically from an annotated corpus and uses statistical methods to assign the most likely parse to a new sentence. The method of 'Data Oriented Parsing' (e.g. [Bod, 1995]) simply uses the annotated corpus itself without abstracting out an explicit grammar. But from a sufficiently abstract point of view, all these systems are doing the same thing: they differ mainly in whether the grammar is represented extensionally as an annotated corpus (or in a convenient representation derived from that), or as an independent intensional object. All systems report good success rates, but have not been compared on the same kinds of data.

It is to be assumed that the appropriate technique for statistical semantics would be more akin to parsing than to translation, even though one's first thoughts might be that assigning a meaning representation is like translating from English to (say) logic. But statistical translation goes from source language to target language word sequences using statistical models derived from aligned bi-lingual corpora. Assigning a meaning is to systematically relate a sentence, component by component, to an expression of a formal language. That formal language must have an articulated enough structure to be able to support the many inferential processes needed for full interpretation of a sentence. It is unlikely that a very articulated representation could be represented as a sequence of symbols and aligned with sentences from some corpus in the way required for statistical translation techniques to be applicable, although coarser grained representations might be treatable in this way.

It may be that when people talk of 'statistical semantics' they mean something like 'semantic tagging'. Several groups (e.g. at Cambridge University Press) have developed systems which select the appropriate sense entry for a word in the context of a sentence, using techniques which are similar to those used for syntactic tagging, or sometimes using the resources of a large online dictionary. In general, it is true to say that all the currently existing instances of what might be called 'statistical semantics' (e.g. [Alshawi and Carter,]) have the same property as this, in that they select between interpretations that are provided by some other mechanism, rather than providing those interpretations themselves by direct statistical means (although e.g. [Dagan and Itai, 1994] have proposed statistical methods for discovering distinct word senses).

What are the prospects, then, for doing 'real' semantics by statistical means? 'Real semantics' here we will assume to consist of two things: firstly, assigning to a sentence something like a 'quasi-logical form', representing the context independent meaning of the sentence; and secondly, relating this QLF to some other representation which makes explicit the contextualised meaning of the sentence.

The only implemented work along these lines that we are aware of is that of the BBN group ([Miller, 1994]) They took a statistical approach to the assignment of QLF-like structure. They annotated a corpus with coarse-grained semantic representations and used Hidden Markov Model techniques to assign meanings to new sentences. This system assigned correct meanings to the test set 85% of the time. (This appears quite impressive, but it is actually less accurate than rule-based systems can achieve on the same data). However, the domain used for these experiments was the ATIS corpus, which is an extremely simple and homogeneous domain to work with. It is an open question whether these techniques will generalise to larger domains, or to richer semantic representations.

It is also not obvious that going directly from strings of words to meanings is the best way to proceed. It might be more effective to use statistical techniques to assign a rich parse tree in a robust way, and then produce meaning representations by rule from such trees: this is a relatively straightforward process.

As regards the second stage of semantic processing mentioned above, we know of no work which uses statistical methods to carry out such tasks. Several groups, however, use methods for pronoun resolution which assume a system of weights of various kinds (e.g. [Lappin and Leass, 1994]). It would be an obvious next step to give a statistical interpretation to these weights. We assume that one reason why these techniques have not been tried out is that there are no training corpora available which contain the right kinds of annotation. When such corpora become widely available we can expect progress in using statistical techniques for assigning semantic representations and will be in a better position to assess whether this is a viable option to pursue.

5.2.9 Dialogue

Most of the successful work in formal semantics over the past twenty years or so which has applications to natural language technology has involved sentence and text processing. While there is recent work in formal semantics which tackles dialogue problems (in DRT, situation semantics and dynamic semantics) it is recent and in need of further development before a useful overview can be given of it. Dialogue processing is currently a major obstacle to the development of advanced robust natural language applications such as dialogic interactions with databases where the use of speech is convenient or essential (e.g. in-car navigation systems) or service applications such as telephonic bank transactions and flight booking. In order to service the needs of such applications as well as other dialogue relate applications which are being worked on such as Verbmobil and dialogue interfaces to route-planning (as currently being developed, for example, at SRI) it will be important in the immediate future to develop formal models of communication in dialogue that incorporate the results of formal semantics at the same time as making explicit how the particular problems of communicating agents are to be treated. Some of the central problems which are being worked on and on which progress might be made in the near future are:

- A characterization of "micro conversational events", i.e. speech events, normally below the level of the sentence, including disfluencies. There is a need to characterize those events which contribute to the content of an utterance and how the content is built up incrementally. The content-contributing events should be separated from those events which are used as a "back-channel", e.g. meta-comments that indicate that the previous word or phrase should be disregarded. There is still a great need for conceptual work which will enable us to understand linguistic communication in terms of speech events processed incrementally rather than the classical view based on the logical notion of an interpreted formal language.

- A characterization of agents' information states appropriate to the treatment of dialogue. Agents form beliefs on the basis of information conveyed in conversation. Not only that, but the way a given contribution to a conversation will be interpreted by a given agent depends on her knowledge, beliefs and goals. This means that the characterization of information states which are relevant to dialogue processing should share something with the kind of intensional structures which are used for the interpretation of attitudinal sentences such as belief-sentences. This connection has been made in recent work but a great deal more effort is needed to create a formal theory which could have practical application.

- An account of updates of agents' information states in relation to dialogue games. In talking about updates in general much work in formal semantics has, of necessity, relied on rather simple notions of information states (e.g. sets of possible worlds) and straightforward notions of update (ever increasing knowledge rather than, for example, belief revision). There is now increasing concern with the fact that this is an oversimplification and that we need more structured notions of information states and more subtle ways of changing them on the basis of a conversational turn. This work is still in its infancy and it would be hard to point to an accepted body of opinion on the best way to achieve this.

- A realistic treatment of communication in dialogue that takes into account the possibility that communication might be quite successful for the purposes at hand without there being an exact match between the information states of the dialogue participants. A practical model of communication in dialogue should be able to say something about how the participants' information states may approximate each other and the associated degrees of successful communication. This is a topic of enormous importance if we are to provide useful theories with a direct application to real dialogue processing. But as yet the problem has been barely formulated.

- A treatment of context that takes into account that in dialogue the reliance on information about the context becomes of even more central importance than in text interpretation, in particular shared or shareable information about the visual and otherwise perceivable environment of the dialogue participants. We know a great deal about where in semantic interpretation there are hooks for a contribution from the context and indeed a traditional view of theoretical semantics is that it should not go further than providing those hooks. One way to build on this work without having to confront the whole general knowledge problem is to build formal models of the way in which information from a visual scene (or other perceived situation) is integrated with linguistic semantic information during the course of a dialogue. Of particular interest is how the visual scene might influence the interpretation of a speech event by an agent.

There is a general perception that one important key to increasing our ability to handle natural dialogue is an increased understanding of the processing of partial or underspecified information. The information available to dialogue participants can be underspecified in several ways. The acoustic signal will not determine what words were uttered. The interpretation will not be fully specified and dialogue participants must reason with underspecified meaning representations in order to determine whether more information should be requested in order to resolve the meaning further or whether the dialogue can be continued for the purposes at hand with the degree of resolution currently available. Information extraction from dialogue by an observer often involves partial semantic processing to an even greater extent since the observer is not always aware of the contextual cues and knowledge that the dialogue participants are relying on.

At the end of this paper we will present a view on information exchange which puts dialogue in proper focus.

5.3 Future Research

In the following two sections we will first take up research tasks which result immediately from our work on FraCaS and then give a more general view of midterm research that could be conducted in computational semantics.

5.3.1 Research Tasks Immediately Resulting from FraCaS

Given the two year length of this project and its large aims we can only pretend to have started research on a number of important areas. We list here some of the things that could be done that build directly on the results we have achieved so far.

Books Arising from the Deliverables

1. A book on different approaches to formal semantics based mainly on the material in D8 and D9 needs to be written

2. A book on a framework for computational semantics based mainly on material in the second year deliverables needs to be written.

   A version of the framework tool might be distributed with one or both books.

Semantic Benchmarks

1. The data presented in Chapter 3 of deliverable D7 need to be honed into a set of benchmarks integrated with the test suite.

2. More work needs to be done rooting the data in real language and we need to explore methods of gathering statistics on the frequency of occurrence of the different phenomena in texts of different genres. This is non-trivial for the most part because we do not yet have good semantic mark-up languages or easy ways of finding semantic phenomena in corpora.

3. Linked to the task of discovering the frequency of semantic phenomena in different genres is the task of discovering what kind of semantic coverage is needed for various classes of application. We need to create benchmarks for particular types of application.

4. All this work on benchmarks needs to be conducted in various languages with a view to discovering where language universal benchmarks can be created and where there need to be benchmarks for phenomena relating to particular languages.

Test Suite

1. The test suite needs to be checked and extended to at least the full coverage of the bench-mark fragment in D7. In those cases where the kind of tests we have proposed are inappropriate, we need to explain why and consider alternatives.

2. The tests need to be rooted in real data. This means finding examples in real texts which correspond to the premises in our tests and where it would be appropriate to draw the inferences indicated by the tests.

3. Sample implementations need to be given which meet the requirements of the tests and which can be integrated as modules within the framework tool.
4. Corresponding tests for different languages need to be developed with a view to seeing how much of the inferencing can be carried over to different languages and illuminating new problems that arise.

Developing the Framework Tool

1. The usability and extendibility of the tool needs to be fully tested by people who have not been involved in its development.
2. The tool as it stands needs to be made available on at least two platforms (X windows, Macintosh).
3. Modules for semantic evaluation in databases of various kinds (e.g. simple relations without negation, simple relations with negation, with quantifiers, with modality, with intensional constructions) need to be developed along with modules illustrating the extent to which the formalisms included in the framework are equivalent with respect to the databases.
4. Model grammars illustrating the description of each of the approaches in the first year deliverables need to be implemented and integrated into the framework tool
5. Modules illustrating the benchmarks and test suite discussed above need to be implemented
6. We need to address the possibility of integrating into the framework tool other recent semantic implementations such as the Dyana Integrated Implementation
7. The relationship between the framework tool and available grammar development tools and formalisms (e.g. CUF, PLEUK, HDRUG) needs to be clarified and possibilities for integration explored
8. We need to build a semantics workbench on the basis of the framework tool. To do this will require extensive discussion with potential users to see what is required over and above the facilities provided by the current toolbox.

5.3.2 A Midterm Perspective – Information exchange in dialogue

A general issue that has not been addressed in the FraCaS project is that of semantics and information exchange in dialogue. This is because theoretical work in this area is just beginning to develop. We see this as being an area of great activity and progress over the next five to ten years both in theoretical and computational semantics. We think that it is important that the theoretical and computational research be conducted side by side and feed each other over this period of development and that the work on information exchange build on the work that has been reviewed in FraCaS rather than starting afresh with a new kind of dialogue semantics and gradually discovering that the tools and techniques discussed in this project are needed to account for information exchange as well as some new ones.

To support this view we give below a view of information exchange which we think fosters the communication between theoretical and practical work and a firm basis on previous semantic work.

Motivation for natural language information exchange technology

Natural language is pervasive in computer systems and is the main means of conveying information to users, either combined with the use of other modalities such as graphics or on its own. Current technology is largely passive with respect to information exchange using natural language – it presents the form of natural language to the user and relies on her understanding of it in order to effect information exchange. There is a need to develop the technology needed to process information represented by natural language in order to facilitate more efficient information exchange with the user by having machines play a more active role in linguistic understanding. Among other things this technology will enable applications that reason about the content of texts and understand and take part in dialogues with the user, in particular dialogues conducted in the modality of speech.

Applications enabled or improved by an understanding of linguistic information exchange include:

1. Natural text to speech (as in machines that read to the blind). Natural prosody relies not only on an understanding of information content but also principles of information exchange associated with the presentation of new and old information.
2. Dialogic interactions with databases where the use of speech is convenient or essential (e.g. in-car navigation systems)
3. Information extraction from large bodies of natural language text (e.g. as is becoming available on the world-wide web). The current technology needs to move from browsers and pattern matchers to intelligent searchers that will be able to enter into dialogues with texts to extract their information.

4. service applications (bank transactions, flight booking)

Some of these application areas are now at the point where they are of commercial significance. For example, phone banking is currently being done by key words and fixed menus and the first group to achieve speech driven phone banking with some dialogue flexibility would gain a great deal. Some applications which are currently of interest could include elements of several of these four general application areas. One example would be home shopping where text to speech could be useful for the reading of catologues and other information concerning products, availability could be reported and sales negotiated by dialogue over the phone, and comparison of products could be carried out by extracting information from a variety of product information and consumer reports. Another example would be tuition and training where natural dialogue could improve the efficiency of automated training systems, increasing the amount of individual tuition available for trainees and relieving teachers of routine tasks. A particularly fruitful area for dialogue technology would be computer assisted language learning (CALL) where the object of the training is for the user to be able to conduct natural dialogue in a foreign language.

Overview of natural language IE Technology

We divide the technology of natural language information exchange into component technologies and then isolate more generic technologies whose development is needed in order to support the components.

Components The component technologies include:

1. Processing of speech with emphasis on the recognition/expression of content
2. Processing of micro conversational events. This includes both the incremental processing of the content of speech events and a way of reasoning about speech events which are used for dialogue control, that is, feedback or back-channel utterances which the dialogue participants use to cooperate on guiding the course of the dialogue.
3. Content extraction from sentential and phrasal events. Deriving information content on the basis of macro-conversational events (sanitized micro-conversational events). This includes reasoning about their communicative function in the dialogue (e.g. their potential role in conversational moves). It involves also non-encyclopedic lexical knowledge (in particular for closed-class items)

4. Dialogue and discourse structures and strategies. This includes reasoning about conversational moves and their role in dialogue management. Important questions concern which aspects of context influence dialogue management. For example, what features of context (including information about the dialogue so far) need to be kept track of and by what mechanisms may utterances be triggered by context?

Supporting technologies The supporting technologies include:

1. Processing underspecified information
2. Processing ill-formed input
3. Interfacing to encyclopedic lexical and general knowledge
4. Processing multi-modal dialogue events (e.g. integrating visual reasoning)
5. Multilingual tuning. The major issue is the extent to which the component technologies need to be tuned for different natural languages. To the greatest extent possible it is desirable to define generic systems which are applicable to all natural languages and which can be further specified to deal with the details of particular languages. Also of interest are technologies to deal with multilingual events such as translation situations.

Bibliography

[Alshawi and Carter,] Alshawi, H. and Carter, D. ter. Training and scaling preference functions for disambiguation. *Computational Linguistics, forthcoming.*

[Black, 1993] Black, E. 1993. Statistically based computer analysis of english. In Black, E.; Garside, R.; and G.Leech, , editors 1993, *Statistically Driven Computer Grammars of English.* Rodopi, Amsterdam.

[Bod, 1995] Bod, R. 1995. The problem of computing the most probable tree in data-oriented parsing. In *Proc European ACL*, Dublin. 104–111.

[Bos *et al.*, 1994] Bos, J.; Mastenbroek, E.; McGlashan, S.; Millies, S.; and Pinkal, M. 1994. A compositional DRS-based formalism for NLP-applications. In *Proceedings of the International Workshop on Computational Linguistics.* University of Tilburg. 21–31.

[Brill, 1994] Brill, E. 1994. A simple rule-based part of speech tagger. In *Proc. Applied Natural Language Processing*, Trento. ACL.

[Briscoe and Copestake, 1995] Briscoe, E.J. and Copestake, A. 1995. Lexical rules in the TDFS framework. Acquilex-II Working Papers, ??

[Briscoe *et al.*, 1990] Briscoe, E.J.; Copestake, A.; and Boguraev, B.K. 1990. Enjoy the paper: Lexical semantics via lexicology. In *13th International Conference on Computational Linguistics (COLING-1990)*, Helsinki. 42–47.

[Briscoe *et al.*, 1995] Briscoe, E.J.; Copestake, A.; and Lascarides, A. 1995. Blocking. In St.Dizier, P.; Viegas, P.; and Viegas, E., editors 1995, *Computational Lexical Semantics.* Cambridge University Press, Cambridge, England.

[Calcagno, 1995] Calcagno, M. 1995. Interpreting lexical rules. Conference on Formal Grammar, Barcelona.

[Carpenter, 1992] Carpenter, B. 1992. *The Logic of Typed Feature Structures.* Cambridge University Press, Cambridge.

[Carter, 1976] Carter, R. 1976. Some constraints on possible worlds. *Semantikos* 1:27–66.

[Cooper *et al.*, 1993] Cooper, Robin; Lewin, Ian; and Black, Alan 1993. Prolog and natural language semantics. Lecture notes for computational semantics, Dept. of Artificial Intelligence, Univ. of Edinburgh.

[Cooper, 1983] Cooper, R. 1983. *Quantification and Syntactic Theory.* Reidel, Dordrecht, Dordrecht.

[Copestake and Briscoe, 1995] Copestake, A. and Briscoe, E.J. 1995. Semi-productive polysemy and sense extension. *Journal of Semantics* 12:15–67.

[Copestake, 1995] Copestake, A. 1995. Representing lexical polysemy. In *Proceedings of the AAAI Spring Symposium on Lexical Semantics*, Stanford, CA.

[Cruse, 1986] Cruse, D.A. 1986. *Lexical Semantics.* Cambridge University Press, Cambridge, England.

[Dagan and Itai, 1994] Dagan, I. and Itai, A. 1994. Word sense disambiguation using a second language monolingual corpus. *Computational Intelligence* 20:563–596.

[Dalrymple *et al.*, 1993] Dalrymple, M.; Lamping, J.; and Sarasawat, V. 1993. LFG semantics via constraiunts. In *Proceedings of the sixth EACL.* 97–105.

[Dowty *et al.*, 1981] Dowty, D.R.; Wall, R.E.; and Peters, S. 1981. *Introduction to Montague Semantics.* Reidel, Dordrecht, Dordrecht.

[Dowty, 1979] Dowty, D.R 1979. *Word Meaning and Montague Grammar.* Reidel, Dordrecht, Dordrecht.

[Dowty, 1989] Dowty, D. R. 1989. On the semantic content of the notion of 'thematic role'. In Partee, B.; Chierchia, G.; and Turner, R., editors 1989, *Properties, Type, and Meanings*, volume 2. Kluwer, Dordrecht. 69–130.

[Evans and Gazdar, 1989a] Evans, R. and Gazdar, G. 1989a. Inference in DATR. In *4th Conference of the European Chapter of the Association for Computational Linguistics (EACL-1989)*, Manchester, England. 66–71.

[Evans and Gazdar, 1989b] Evans, R. and Gazdar, G. 1989b. The semantics of DATR. In Cohn, A.G., editor 1989b, *Proceedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and Simulation of Behavior (AISB)*, London.

[Fillmore, 1968] Fillmore, C. 1968. The case for case. In Bach, E. and Harms, T., editors 1968, *Universals in Linguistic Theory.* Holt, Rinehart, Williams, New York.

[Frank and Reyle, 1995] Frank, A. and Reyle, U. 1995. Principle-based semantics for hpsg. In *7th Conference of the European Chapter of the Association for Computational Linguistics (EACL-1995)*, Dublin. 9–16.

[Frege, 1879] Frege, G. 1879. *Begriffsschrift, eine der arithmetischen nachgebildete Formelprache des reinen Denkens.* Verlag Nebert, Halle.

[Gazdar *et al.*, 1985] Gazdar, G.; Klein, E.; Pullum, G.; and Sag, I. 1985. *Generalized Phrase Structure Grammar.* Basil Blackwell, Oxford.

[Hobbs *et al.*, 1992] Hobbs, J.R.; Stickel, M.; Appelt, D.; and Martin, P. 1992. Interpretation as Abduction. *Artificial Intelligence Journal.*

[Hudson, 1995] Hudson, R. 1995. Identifying the linguistic foundations for lexical research and dictionary design. In Walker, D.; Zampolli, A.; and Calzolari, N., editors 1995, *Automating the Lexicon: Research and Practice in a Multilingual Environment.* Oxford University Press, Oxford. First published 1986.

[Kamp and Reyle, 1993] Kamp, H. and Reyle, U. 1993. *From Discourse to Logic.* Kluwer, Dordrecht.

[Karlsson, 1995] Karlsson, F. et al 1995. *Constraint Grammar.* Mouton de Gruyter.

[Keller, 1988] Keller, W.R. 1988. Nested Cooper storage. In Reyle, U. and Rohrer, C., editors 1988, *Natural Language Parsing and Linguistic Theory.* D. Reidel, Dordrecht, The Netherlands. 432–447.

[Konrad, 1996] Konrad, Karsten 1996. Extending CLIG: Interaction and user defined graphics. Technical report, Universität des Saarlandes. forthcoming Claus Report.

[Kuschert, 1995] Kuschert, Susanna 1995. A type-theoretic semantics for λ-drt. *Proceedings of the 10th Amsterdam Colloquium.* forthcoming.

[Lappin and Leass, 1994] Lappin, S. and Leass, H. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics* 20:535–562.

[Lascarides and Asher, 1993] Lascarides, A. and Asher, N. 1993. Temporal interpretation, discourse relations and common sense entailment. *Linguistics and Philosophy* 16:437–493.

[Lascarides and Copestake, 1995] Lascarides, A. and Copestake, A. 1995. A logic for order-independent typed default unification. Acquilex-II Working paper 60.

[Lascarides *et al.*, 1994] Lascarides, A.; Briscoe, E.J.; Asher, N.; and Copestake, A. 1994. Persistent associative default unification. Linguistics and Philosophy, in press.

[Lascarides *et al.*, 1996] Lascarides, A.; Copestake, A.; and Briscoe, E.J. 1996. Ambiguity and coherence. Journal of Semantics, in press.

[Lascarides, 1995] Lascarides, A. 1995. The pragmatics of word meaning. In *Proceedings of the AAAI Spring Symposium on Lexical Semantics*, Stanford, CA.

[Levin, 1993] Levin, B. 1993. *Towards a Lexical Organisation of English Verbs.* University of Chicago Press, Chicago.

[Levin, 1995] Levin, B. 1995. Approaches to lexical semantic representation (lexical semantics in review: an introduction). In Walker, D.; Zampolli, A.; and Calzolari, N., editors 1995, *Automating the Lexicon: Research and Practice in a Multilingual Environment.* Oxford University Press, Oxford. First published 1985.

[Maier *et al.*, 1996] Maier, Holger; Milward, David; and Pinkal, Manfred 1996. An education and research tool for computational semantics. Technical report, Universität des Saarlandes. forthcoming Claus Report.

[Miller, 1994] Miller, S. et al. 1994. Statistical language processing using hidden understanding models. In *ARPA Workshop on Human Language Technology,* Plainsboro. 266–270.

[Moens and Steedman, 1988] Moens, M. and Steedman, M. 1988. Temporal ontology and temporal reference. *Computational Linguistics* 14(2):15–28.

[Montague, 1973] Montague, R. 1973. The proper treatment of quantification in ordinary English. In e.a., J. Hintikka, editor 1973, *Approaches to Natural Language.* Reidel. 221–242.

[Moravsik, 1975] Moravsik, J. 1975. Aitia as generative factor in Aristotle's philosophy. *Dialogue* 14:622–636.

[Muskens, 1994] Muskens, R. 1994. A compositional discourse representation theory. In Dekker, P. and Stokhof, M., editors 1994, *Proceedings 9th Amsterdam Colloquium.* ILLC, Amsterdam. 467–486.

[Ousterhout, 1994] Ousterhout, J. 1994. *Tcl and the Tk Toolkit.* Professional Computing. Addison-Wesley, Reading, Massachusetts.

[Pinker, 1989] Pinker, S. 1989. *Learnability and Cognition: The Acquisition of Argument Structure.* MIT Press, Cambridge, MA.

[Pollard and Sag, 1987] Pollard, C. and Sag, I. 1987. *An Information-Based Approach to Syntax and Semantics: Volume 1, Fundamentals.* CSLI Lecture Notes 13. CSLI, Stanford. Distributed by University of Chicago Press.

[Pustejovsky and Boguraev, 1993] Pustejovsky, J. and Boguraev, B. 1993. Lexical knowledge representation and natural language processing. *Artificial Intelligence* 63:193–223.

[Pustejovsky and Busa, 1995] Pustejovsky, J. and Busa, F. 1995. Unaccusativity and event composition. In Bertinetto, P.; Binachi, V.; Higginbotham, J.; and Squartini, M., editors 1995, *Temporal Reference: Aspect and Actionality.* Rosenberg and Sellier, Turin.

[Pustejovsky, 1991] Pustejovsky, J. 1991. The generative lexicon. *Computational Linguistics* 17(4):409–441.

[Pustejovsky, 1995] Pustejovsky, J. 1995. *The Generative Lexicon.* MIT Press, Cambridge, MA.

[Reyle, 1993] Reyle, U. 1993. Dealing with ambiguities by underspecification: Construction, representation and deduction. *Journal of Semantics* 10:123–179.

[Riehemann, 1993] Riehemann, S. 1993. Word formation in lexical type hierarchies. Master's thesis, University of Tübingen, Germany.

[SICStus, 1995] SICStus, 1995. *SICStus Prolog User's Manual.* Swedish Institute of Computer Science, Kista.

[Zaenen, 1993] Zaenen, A. 1993. Unaccusativity in Dutch: The interface of syntax and lexical semantics. In Pustejovsky, J., editor 1993, *Semantics and the Lexicon.* Kluwer, Dordrecht.