An Education and Research Tool for Computational Semantics

Karsten Konrad[1], Holger Maier[1], David Milward[2] and Manfred Pinkal[1]

(1) Computerlinguistik,
Universität des Saarlandes
66041 Saarbrücken, Germany
konrad, maier, pinkal@coli.uni-sb.de

(2) SRI International,
Suite 23, Millers Yard
Cambridge, CB2 1RQ, GB
milward@cam.sri.com

Abstract

This paper describes an interactive graphical environment for computational semantics. The system provides a teaching tool, a stand alone extendible grapher, and a library of algorithms together with test suites. The teaching tool allows users to work step by step through derivations of semantic representations, and to compare the properties of various semantic formalisms such as Intensional Logic, DRT, and Situation Semantics. The system is freely available on the Internet.

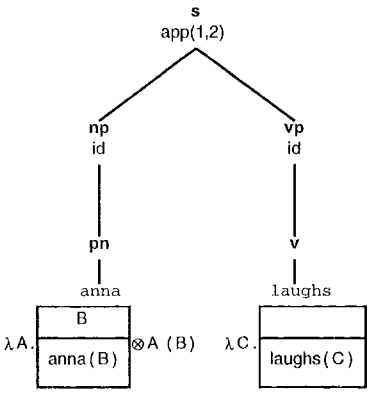


Figure 1: Initial Representation of `Anna laughs` with λ-DRT

1 Introduction

The CLEARS tool (Computational Linguistics Education and Research Tool in Semantics) was developed as part of the FraCaS project[1] which aimed to encourage convergence between different semantic formalisms. Although formalisms such as Intensional Logic, DRT, and Situation Semantics look different on first sight, they share many common assumptions, and provide similar treatments of many phenomena. The CLEARS 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.

The first part of the paper shows the potential of the system for investigating the properties of different semantic formalisms, and for teaching students formal semantics. The next section outlines the library contents and the system architecture, which was designed to reflect convergence between theories. The result is a highly modular and, we believe, a highly flexible system which allows user programs to be integrated at various levels. The final part of the paper describes the grapher which was designed as a stand alone tool which can be used by various applications.

[1]A Framework for Computational Semantics, European Community LRE 62-051.

2 A Tutorial System for Computational Semantics

As a tutorial tool, CLEARS allows students to investigate certain formalisms and their relationship. It also provides the possibility for the teacher to provide interactive demonstrations and to produce example slides and handouts.

In this section we show how a user can interactively explore the step-by-step construction of a semantic representation out of a syntax tree. Figures 1 and 2 show a possible initial display for the sentence "Anna laughs" in a compositional version of DRT (Bos et al., 1994) and in 'Montague Grammar' (Dowty et al., 1981).

The user controls the semantic construction process by moving to particular nodes in the derivation tree, and performing operations by using mouse double-clicks, or by selecting from a pop-up menu. For example, clicking on **app(2,1)**

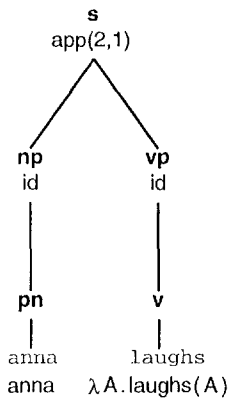


Figure 2: Initial Representation of `Anna laughs` with 'Montague Grammar'

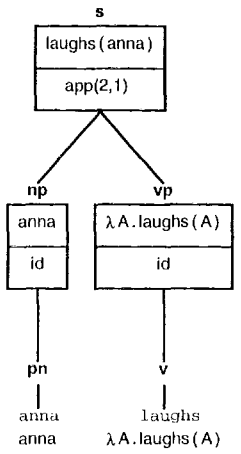


Figure 3: Final Representation of `Anna laughs` in 'Montague-Grammar'

in the tree shown in Figure 2 has the effect of applying the lambda-expression **λA.laughs(A)** to **anna**. The resulting display is given in Figure 3.

The pop-up menu allows a user to perform single derivation steps. For example, the user can first form an application term **λA.laughs(A)(anna)** and then reduce this at the next step. Menu options include the possibility of cancelling intensional operators, performing lambda reduction, applying meaning postulates, and DRS merging. The menu also allows a user to choose whether or not to perform quantifier storage or discharge, and thereby pick a particular reading for a sentence. Alternatively the user can choose to fully process a node, in which case all readings are simultaneously displayed.

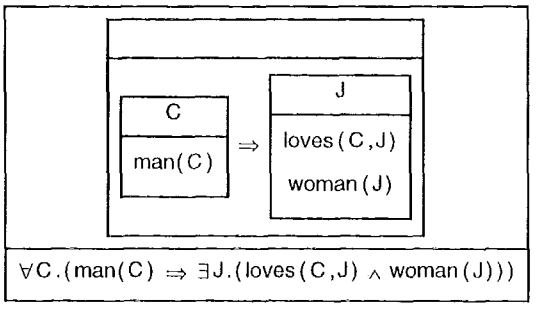


Figure 4: Translating DRT to Predicate Logic

3 Comparing Theories

A major use of the tool is for comparison of different semantic theories and methods of semantic construction. To aid comparison of theories, there are translation routines between some semantic formalisms. For example, Figure 4 shows a translation from a DRS to a formula in Predicate Logic.

The user can try out various options for semantic construction by using a menu to set various parameters. An illustrative subset of the parameters and their possible values is given below:

semantic formalism
Logic of Generalized Quantifiers,
Intensional Logic,
Compositional DRT (Muskens, 1993),
λ-DRT (Bos et al., 1994),
Top-Down-DRT (Kamp and Reyle, 1993),
Situation Semantics.

grammar
simple PSG, PSG with features,
Categorial Grammar with features.

parser
top-down, incremental (for CG only).

lexicon
simple lexicon, lexicon with features.

syntax-semantics mapping
rule-to-rule, syntactic template.

syntax-semantics construction
serial, parallel.

subject applied to verb phrase
yes, no.

quantifier storage mechanism
Cooper Storage (Cooper, 1983),
Nested Cooper Storage (Keller, 1988)

β-reduction
unification based, substitution based.

4 The Library

Because a tutorial system of this kind has to be based largely on standard routines and algorithms that are fundamental for the area of computational semantics, a secondary aim of the project was to provide a set of well documented programs which could form the nucleus of a larger library of reusable code for this field. Most of the library contents correspond directly to particular values of parameter settings. However there are some extra library routines, for example a very generalised form of function composition. The library is being expanded with routines for semantic construction driven by semantic types. It is also intended to integrate a wider range of grammars, parsing strategies and pronoun resolution strategies. For program documentation we largely have followed the approach taken in LEDA (Näher, 1993)).

Apart from the routines concerned directly with computational semantics, there are also routines designed to aid application developers who want to provide a graphical output for semantic representations. These routines are mainly concerned with translating from Prolog syntax into the description string syntax used by the *CLiG* grapher. Currently they rely on the Tcl/Tk library package provided by Sicstus 3.

4.1 Modularisation Principles

A standard approach to modularisation is to split a problem into independent *black boxes*, e.g. a grammar, a parser etc. This top-down modularisation is then followed by some bottom-up modularisation in the sense of supplying general utilities which each of the larger modules can use. For this application, such an approach had obvious inadequacies. For example, there are subtle differences in some steps of quantifier storage according to the formalism being used, similarly, differences even in lambda reduction (for intensional logic it is natural to interleave the step of operator cancellation between β-reductions). Even the parsing stage cannot be totally independent unless we generalise to the worst case (the Situation Semantics fragment requires an utterance node as well as a sentence node).

One of the aims in building the tool was to show where semantic formalisms converge. Thus there was theoretical motivation to ensure components of the system were shared wherever possible. There was also practical motivation, since there is more chance of finding errors in shared code. The solution adopted was to use *parameterised modularisation*. This allows differences to be located in as small pieces of code as possible (e.g. single lines of the quantifier storage routine), with the parameters picking up the correct piece of code at run time. There are some small costs due to indirection (instead of calling e.g. a β-reducer directly, a program first calls a routine which chooses the β-reducer according to the parameters). But with these parameterisation layers we provide natural points where the system can be extended or modified by the user. The approach also gets rid of the need to create large data structures which include information which would be relevant for one choice of parameters, but not the current choice. For example, in parsing, a parameterised level chooses how to annotate nodes so that the syntax trees only have the relevant information for the chosen syntax-semantics strategy. The architecture is illustrated in Figure 5.

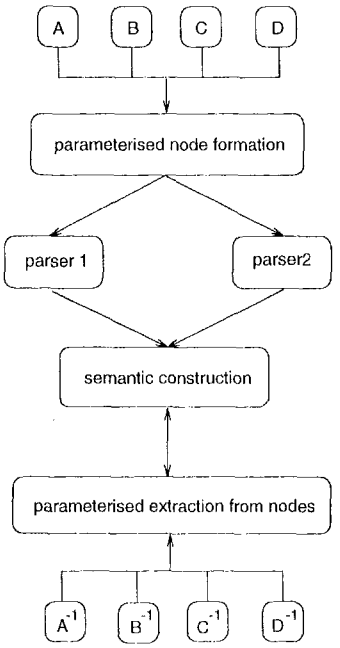


Figure 5: Architecture of a part of the Syntax-Semantics Interface

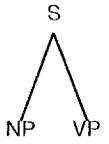
The result of the parameterised approach is a system which provides several thousand possible valid combinations of semantic formalism, grammar, reducer etc. using a small amount of code.

5 The Graphical Interface

A major part of our work on the educational tool was the development of a general graphical browser or *grapher* for the graphical notations used in computational linguistics, especially those in computational semantics such as trees, Attribute-Value-Matrices, EKN (Barwise and Cooper, 1993) and DRSs. The grapher was

written in Tcl/Tk, a programming system for developing graphical user interfaces (Ousterhout, 1994). Two attributes of Tcl/Tk which were important for this application were the provision of translation routines from graphic canvasses into Postscript (allowing generation of diagrams such as Figures 1 to 4), and the ease of providing scaling routines for zooming.

The grapher was designed to be extendible for future applications. Graphical structures are described using a *description string*, a plain text hierarchical description of the object to be drawn without any exact positioning information. For example, the following tree:



is created by the description string:

```
{tree {plain-text "S"}
      {plain-text "NP"}
      {plain-text "VP"}}
```

CLIG can display *interactive* graphical structures which allow the user to perform actions by clicking on mouse-sensitive regions in the display area. The grapher and an underlying application therefore can behave in a way that the grapher is not only a way to visualise the data of the application, but also provides a real interface between user and application.

6 Availability of the System

The system currently requires Sicstus 3 plus Tcl version 7.4 and Tk version 4.0 (or later versions). It is available at the ftp address: `ftp.coli.uni-sb.de:/pub/fracas` or on the WWW at the URL:

`http://coli.uni-sb.de/~clears/clears.html`

Further documentation of the system is given in (FraCaS, 1996a) and (FraCaS, 1996b), which are available from:

`http://www.cogsci.ed.ac.uk/~fracas/`

7 Conclusion

Initial reactions to demonstrations of the educational tool suggest that it has the potential to become a widely used educational aid. We also believe that the programs implemented and documented in this work provide the nucleus of a larger library of reusable programs for computational semantics. Our current plans are to test the system with a wide class of users to discover areas requiring extension or modification. A longer term aim is to integrate the system with existing grammar development environments.

Acknowledgements

This work would not have been possible without the encouragement and support of the other members of the FraCaS Project. We would especially like to thank Robin Cooper, Massimo Poesio and Steven Pulman for contributions to the code.

References

J. Barwise and R. Cooper. 1993. Extended Kamp notation. In Y. Katagiri P. Aczel, D. Israel and S. Peters, editors, *Situation Theory and its Application Vol. 3*, chapter 2, pages 29–54. CSLI, Stanford.

J. Bos, E. Mastenbroek, S. McGlashan, S. Millies, and M. Pinkal. 1994. A compositional DRS-based formalism for nlp-applications. In *Proceedings of the International Workshop on Computational Semantics*, pages 21–31, Tilburg.

R. Cooper. 1983. *Quantification and Syntactic Theory*. SLAP. Reidel, Dordrecht.

D. Dowty, R. Wall, and S. Peters. 1981. *Introduction to Montague Semantics*. SLAP. Reidel, Dordrecht.

FraCaS. 1996a. Building the framework. Fracas Deliverable D15.

FraCaS. 1996b. Using the framework. Fracas Deliverable D16.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic*. Kluwer, Dordrecht.

W. Keller. 1988. Nested cooper storage. In U. Reyle and C. Rohrer, editors, *Natural Language Parsing and Linguistic Theories*, pages 432–447. Reidel, Dordrecht.

R. Muskens. 1993. A compositional discourse representation theory. In P. Dekker and M. Stokhof, editors, *Proceedings of the 9th Amsterdam Colloquium*, pages 467–486. ILLC, University of Amsterdam.

S. Näher. 1993. Leda manual version 3.0. Technical Report MPI-I-93-109, Max-Planck-Institut für Informatik, Saarbrücken, February.

J. Ousterhout. 1994. *Tcl and the Tk Toolkit*. Professional Computing. Addison-Wesley, Reading, Massachusetts.