# Hdrug. A Flexible and Extendible Development Environment for Natural Language Processing.

**Gertjan van Noord**
**Gosse Bouma**

Alfa-informatica & BCN,
University of Groningen
{vannoord,gosse}@let.rug.nl

## Abstract

Hdrug is an environment to develop grammars, parsers and generators for natural languages. The package is written in Sicstus Prolog and Tcl/Tk. The system provides a graphical user interface with a command interpreter, and a number of visualisation tools, including visualisation of feature structures, syntax trees, type hierarchies, lexical hierarchies, feature structure trees, definite clause definitions, grammar rules, lexical entries, and graphs of statistical information of various kinds.

Hdrug is designed to be as flexible and extendible as possible. This is illustrated by the fact that Hdrug has been used both for the development of practical real-time systems, but also as a tool to experiment with new theoretical notions and alternative processing strategies. Grammatical formalisms that have been used range from context-free grammars to concatenative feature-based grammars (such as the grammars written for ALE) and non-concatenative grammars such as Tree Adjoining Grammars.

## 1 Introduction

Hdrug is an environment to develop grammars, parsers and generators for natural languages. The system provides a number of visualisation tools, including visualisation of feature structures, syntax trees, type hierarchies, lexical hierarchies, feature structure trees, definite clause definitions, grammar rules, lexical entries, and graphs of statistical information e.g. concerning cputime requirements of different parsers. Visualisation can be requested for various output formats, including ASCII text format, TK Canvas widget, LaTeX output, and CLiG output (Konrad et al., 1996).

Extendibility and flexibility have been major concerns in the design of Hdrug. The Hdrug system provides a small core system with a large library of auxiliary relations which can be included upon demand. Hdrug extends a given NLP system with a graphical user interface and a number of visualisation tools. Applications using Hdrug typically add new features on top of the functionality provided by Hdrug. The system is easily extendible because of the use of the Tcl/Tk scripting language, and the availability of a large set of libraries. Flexibility is obtained by a large number of global flags which can be altered easily to change aspects of the system. Furthermore, a number of hook predicates can be defined to adapt the system to the needs of a particular application.

The flexibility is illustrated by the fact that Hdrug has been used both for the development of grammars and parsers for practical systems (Boves et al., 1995; van Noord et al., 1996), but also as a tool to experiment with new theoretical notions and alternative processing strategies, such as those discussed by (Carpenter, 1992), (van Noord and Bouma, 1994), (van Noord, 1994). Furthermore, Hdrug has been used extensively both for batch processing of large text corpora, and also for demonstrating particular applications for audiences of non-experts.

Hdrug is implemented in SICStus Prolog version 3, exploiting the built-in Tcl/Tk library. The Hdrug sources are available free of charge under the Gnu Public Licence copyright restrictions. Further information, including the sources and an on-line manual, is available on the World Wide Web. [1]

In this paper we illustrate the functionality of Hdrug, and its extendible and flexible nature, by

[1] The URL is:
`http://www.let.rug.nl/~vannoord/Hdrug/`

means of two examples: ALE and OVIS.

## 2 Overview

This section gives an overview of the functionality provided by Hdrug.

### 2.1 Interface

Hdrug provides three ways of interacting with the underlying NLP system:

- Using an extendible command interpreter.
- Using Prolog queries.
- Using an extendible graphical user interface (based on Tcl/Tk).

The first two approaches are mutually exclusive: if the command interpreter is listening, then you cannot give ordinary Prolog commands and vice versa. In contrast, the graphical user interface (with mouse-driven menu's and buttons) can always be used. This feature is very important and sets Hdrug apart from competing systems. It implies that we can use at the same time the full power of the Prolog prompt (including tracing) and the graphical user interface. Using the command interpreter (with a history and alias mechanism) can be useful for experienced users, as it might be somewhat faster than using the mouse (but note that many menu options can be selected using accelerators). Furthermore, it is useful for situations in which the graphical user interface is not available (e.g. in the absence of an X workstation). The availability of a command-line interface in combination with mouse-driven menu's and buttons illustrates the *flexible* nature of the interface.

An important and interesting property of both the command interpreter and the graphical user interface is *extendibility*. It is very easy to add further commands (and associated actions) to the command interpreter (using straightforward DCG syntax). The graphical user interface can be extended by writing Tcl/Tk scripts, possibly in combination with some Prolog code. A number of examples will be given in the remainder of this paper.

Finally note that it is also possible to run Hdrug without the graphical user interface present (simply give the `-notk` option at startup). This is sometimes useful if no X workstation is available (e.g. if you connect to the system over a slow serial line), but also for batch processing. At any point you can start or stop the graphical user interface by issuing a simple command.

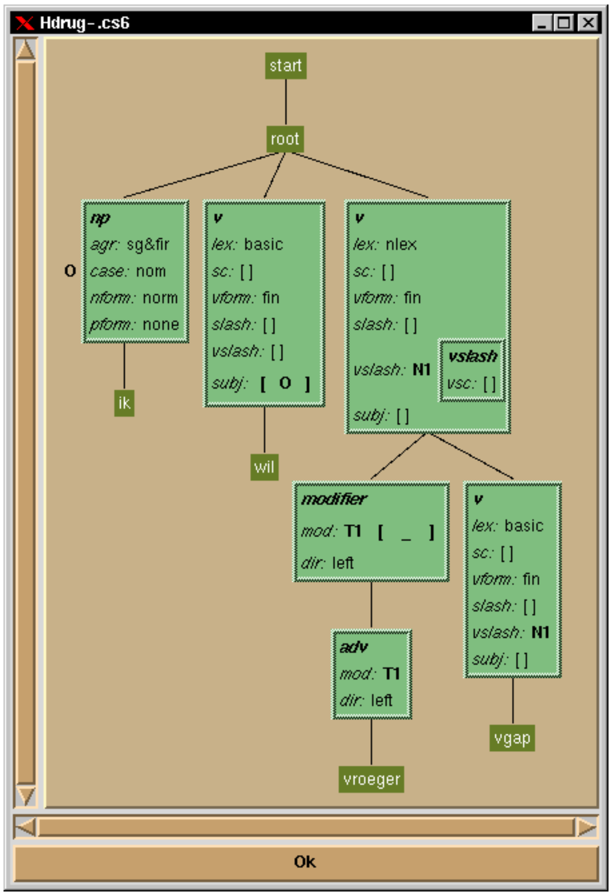



Figure 1: Example of visualisation provided by Hdrug. In this example the derivation tree for the sentence ‘ik wil vroeger’ (*I want earlier*) is shown in a TK widget.

### 2.2 Visualisation

Hdrug supports the visualisation of a large collection of data-structures into a number of different formats.

These formats include [2]:

- ASCII text
- Tk Canvas
- LaTeX
- CLiG

The Tk Canvas format is the format best integrated with the graphical user interface. The data-structures for which visualisation is provided are:

[2]At the moment not all datastructures are supported for all formats. For example, plots of two dimensional data is only available for Tk.

- Trees. Various tree definitions can exist in parallel. For example, the system supports the printing of syntax trees, derivation trees, type hierarchy trees, lexical hierarchies etc. Actions can be defined which are executed upon clicking on a node of a tree. New tree definitions can be added to the system by simple declarations.

- Feature structures. Clicking on attributes of a feature-structure implode or explode the value of that attribute. Such feature structures can be the feature structures associated with grammar rules, lexical entries, macro definitions and parse results.

- Trees with feature structure nodes. Again, new tree definitions can be declared. An example is provided in figure 1.

- Graph (plots of two variable data), e.g. to display the (average) cputime or memory requirements of different parsers.

- Tables.

- Prolog clauses.

- Definite clauses with feature structure arguments. This can be used e.g. to visualise macro definitions, lexical entries, and grammar rules (possibly with associated constraints).

### 2.3 Parser and Generator Management

Hdrug provides an interface for the definition of parsers and generators. Hdrug manages the results of a parse or generation request. You can inspect these results later. Multiple parsers and generators can co-exist. You can compare some of these parsers with respect to speed and memory usage on a single example sentence, or on sets of pre-defined example sentences. Furthermore, actions can be defined which are executed right before parsing (generation) starts, or right after the construction of each parse result (generation result), or right after parsing is completed. For example, in the ALE system to be discussed in the next section, a parse-tree is shown automatically for each parse result. As another example, for the OVIS system discussed in section 4, a word graph is read-in in an ASCII buffer and converted to an appropriate Prolog format before parsing starts.

### 2.4 Useful libraries

Most of the visualisation tools are available through libraries as well. In addition, the Hdrug library contains mechanisms to translate Prolog terms into feature structures and vice versa (on the basis of a number of declarations). Furthermore, a library is provided for the creation of 'Mellish' Prolog terms on the basis of boolean expressions over finite domains (Mellish, 1988). The reverse translation is provided too. Such terms can be used as values of feature structures to implement a limited form of disjunction and negation by unification.

A number of smaller utilities is provided in the library as well, including libraries which extend term_expansion, an add_clause mechanism (based on chapter 9.1 of (O'Keefe, 1990)), management of global variables (the predicate `flag/3` from (Ross, 1989)), support for debugging, etc.

### 2.5 Example Applications

A number of example applications is included in the Hdrug distribution.

- ALE (Carpenter, 1992), including the example HPSG grammar and CG grammar. Adding other ALE grammars is trivial.

- Definite-clause Grammar (Pereira and Warren, 1980) for Dutch to illustrate semantic-head-driven generation (Shieber et al., 1989), and to compare different parsers for speed (Bouma and van Noord, 1993).

- Constraint-based Categorial Grammar, with delayed evaluation of constraints (Bouma and van Noord, 1994).

- HPSG with lexical rules as delayed constraints (van Noord and Bouma, 1994).

- Head-driven Parsing for Tree Adjoining Grammars, as described in (van Noord, 1994)

- A few toy grammars in the Extraposition Grammar formalism (Pereira, 1981).

## 3 ALE

To illustrate the functionality of Hdrug we use Bob Carpenter and Gerald Penn's ALE system (Carpenter, 1992). To quote the authors:

> ALE is an integrated phrase structure parsing and definite clause logic programming system in which the terms are typed feature structures. Typed feature structures combine type inheritance and appropriateness specifications for features and their values. The feature structures used in ALE generalize the common feature structure systems found in the linguistic programming

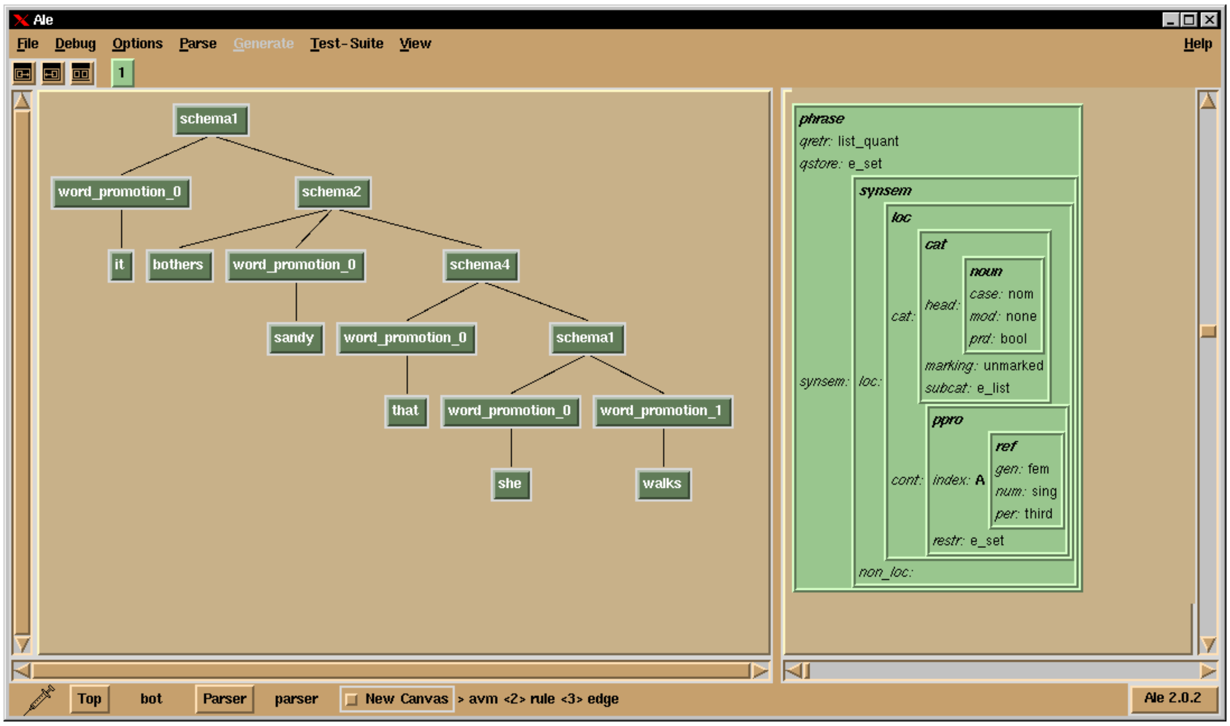



Figure 2: Main Hdrug window for ALE. The nodes of the derivation tree can be clicked to obtain the associated feature structure in the right-most canvas. By clicking on attributes of the feature structures it is very easy to implode and explode parts of feature structures to concentrate on those parts of particular importance for the user. The VIEW menu provides an interface to the visualisation of all ALE datastructures including (lexical) rules, macro's, definite clauses, lexical entries, and edges of the chart.

> systems PATR-II and FUG, the grammar formalisms HPSG and LFG, as well as the logic programming systems Prolog-II and LOGIN. Programs in any of these languages can be encoded directly in ALE.

Because ALE is available for SICStus Prolog, and because ALE only provides a very limited user interface, it provides a particular simple and useful example of an application for Hdrug. The combined ALE/Hdrug system consists of the original ALE sources plus about 450 lines of Prolog code and 250 lines of Tcl code. These define the interface to Hdrug and provide some useful extensions to the graphical user interface. Apart from this, any specific ALE grammar further specifies a small number of declarations. For the example HPSG grammar which is included in the ALE distribution (a rather large grammar: 1650 lines of ALE code) this required only 8 lines of Prolog code. The following examples assume the HPSG example grammar.

Figure 2 shows the main Hdrug window after loading the ALE system with the HPSG grammar and after the parse of the example sentence `she sees a book`.

The Hdrug window consists of two large canvases which are used to display important data-structures. In this case the left-most canvas displays the derivation tree of one of the analyses of the example sentence and the right-most canvas displays the feature structure containing the semantic representation of the top-node of one of the parse results. Immediately under the menu-bar a sequence of buttons is displayed which are labelled '1' and '2'. These represent the results of parsing. If such a button is pressed a pull-down menu is displayed which allows the user to visualise that particular result of the parser in one of the available formats. For example, it is possible to inspect the parse tree of this object, where each node of the tree is a feature structure (the result would be to large to be displayed in a readable form here). Note that it is also possible to obtain a visualisation of the feature structure associated with the top-most node of the parse tree in a specific format. These formats include a straightforward interface to ALE's built-in pretty print routines.

The menu-bar provides an interface to many of the standard functions of Hdrug. The FILE menu-

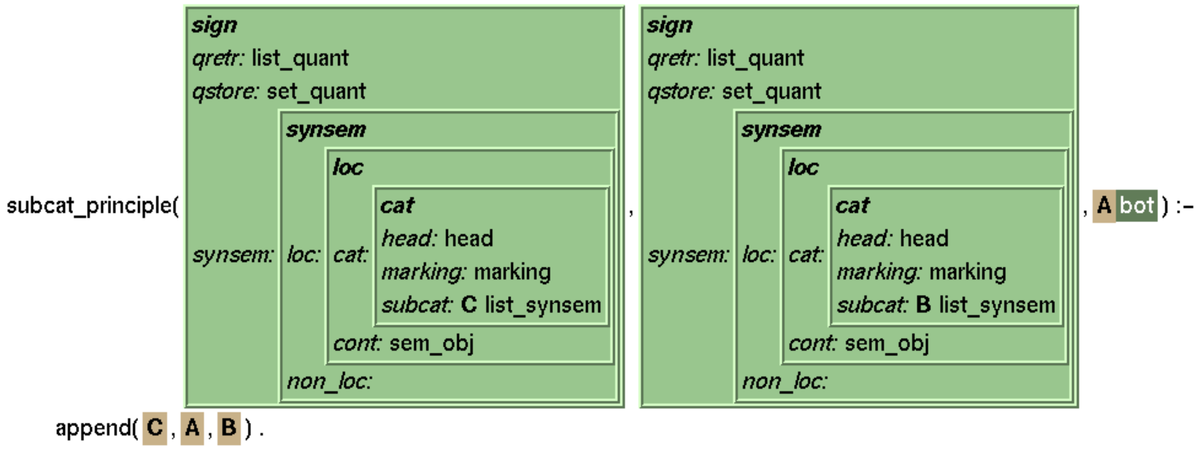



Figure 3: Display of the Ale definite clause definition of the subcat principle.

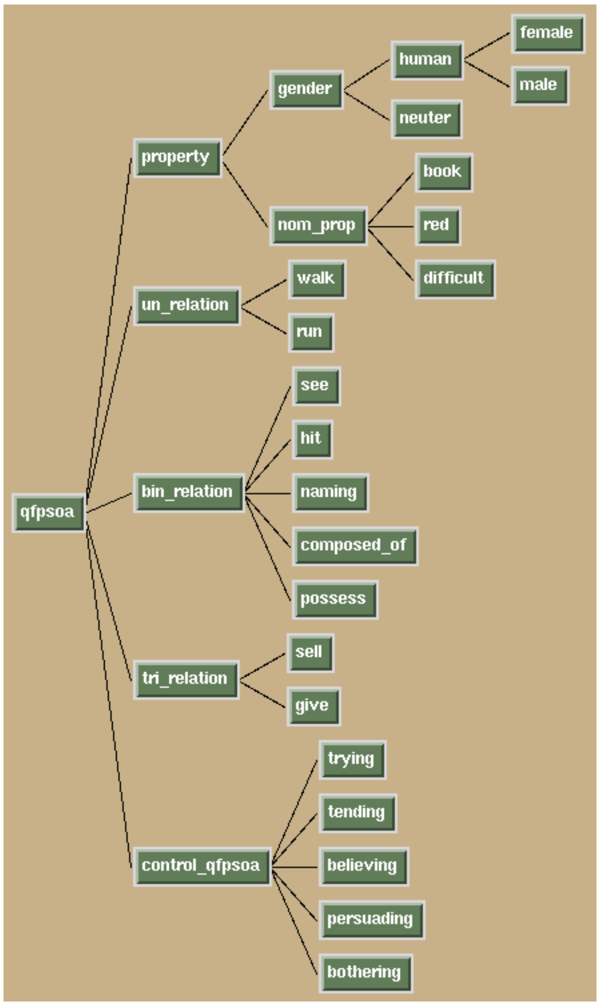



Figure 4: Visualisation of part of the type signature of the HPSG grammar distributed with ALE.

button includes options to load grammar files, Prolog files and Tcl/Tk files. The OPTIONS menu provides an interface to global Hdrug variables. Such variables include the value of the top-category for parsing (the start symbol); the default parser; whether or not the system should check if an object is created whether such an object already exists (this feature is used to recognize spurious ambiguities), etc. The PARSE and GENERATE menu buttons are straightforward means to parse a sentence or to generate a sentence for a given logical form. Note that ALE does not provide a generator, so this menu-button is inactive. If a parse is requested a dialog box is displayed in which you can choose a sentence from a predefined set of example sentences, or in which you can type in a new sentence.

The VIEW menu-button is associated with a pull-down menu which is specific to the Ale application. It provides an interface to visualisation routines for the following important ALE datastructures:

- Edges of the chart
- Lexical entries
- Macro definitions
- Phrase structure rules
- Lexical rules
- Types
- Empty Categories
- Definite Clauses
- Type Signature

For example, the `subcat_principle/3` relation is displayed as in Figure 3.

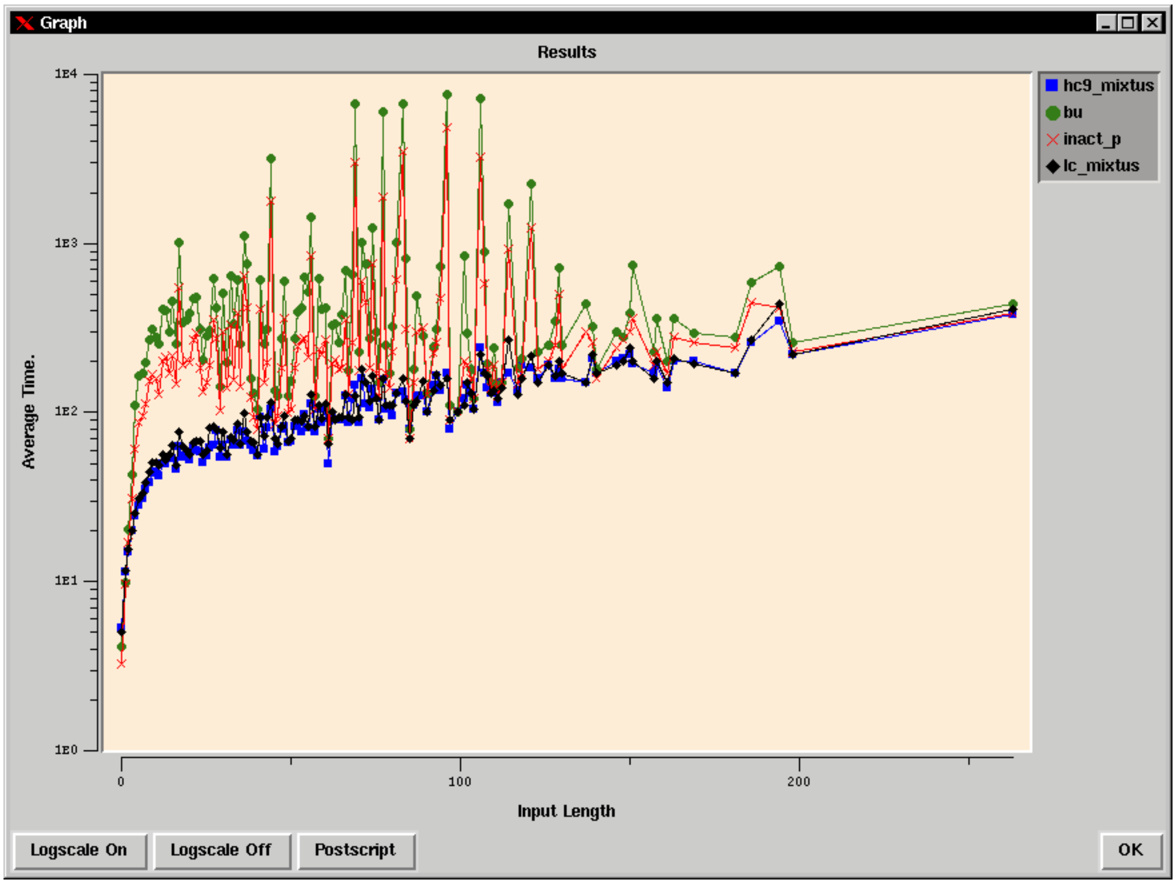



Figure 5: Example of Hdrug support for comparison of different parsers for the same grammar and test sets. In this example a left-corner (lc_mixtus) parser, a head-corner (hc9_mixtus) parser, an inactive chart parser (inact_p) and a bottom-up active chart parser (bu) were compared on a test-set of 5000 word graphs. Timings are in milliseconds and the input size is the number of transitions in the word graph. Note that in this example the parsers only parse the best path through the word graph. The left-corner and head-corner parsers perform this task much faster than the other two: average CPU-times are up to 500 milliseconds, whereas the chart-based parsers require up to 8000 milliseconds on average.

## 4 OVIS

The NWO Priority Programme *Language and Speech Technology* is a research programme aiming at the development of spoken language information systems. Its immediate goal is to develop a demonstrator of a public transport information system, which operates over ordinary telephone lines. This demonstrator is called OVIS, Openbaar Vervoer Informatie Systeem (*Public Transport Information System*). The language of the system is Dutch. Refer to (Boves et al., 1995; van Noord et al., 1996) for further information of this Programme.

The natural language understanding component of OVIS analyses the output of the speech recognizer (a *word graph*) and passes this analysis to the dialogue manager (as an *update expression*). Word graphs are weighted acyclic finite-state automata which represent in a compact format the hypotheses of a speech recognizer. Each path through the word graph is a possible analysis of the user utterance; weights indicate the confidence of the speech recognizer.

The relation between such word graphs and update expressions is defined by means of a Definite Clause Grammar of Dutch. This DCG and a number of parsers have been developed with the Hdrug system. The functionality of Hdrug has been used to compare the different parsers with respect to efficiency on sets of sentences and word graphs. For example, upon loading a specific set of such word graphs, the system can be asked to parse each of the word graphs with a specified subset of the available parsers, and to display information concerning parse

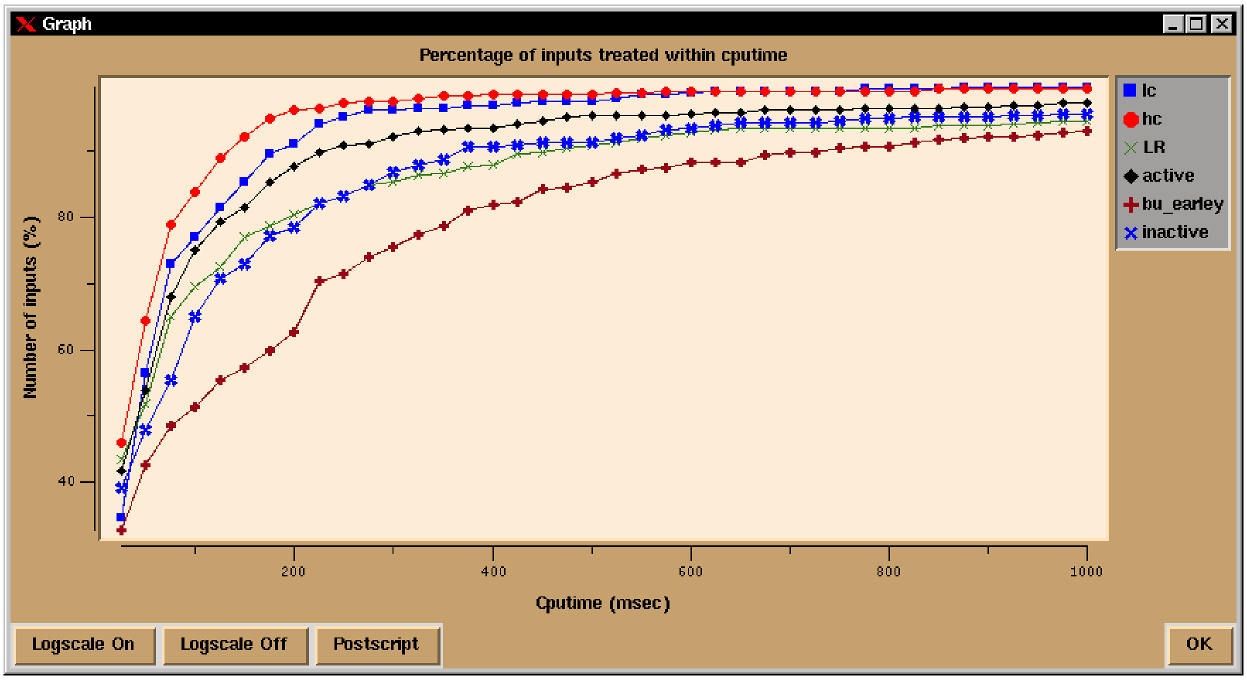


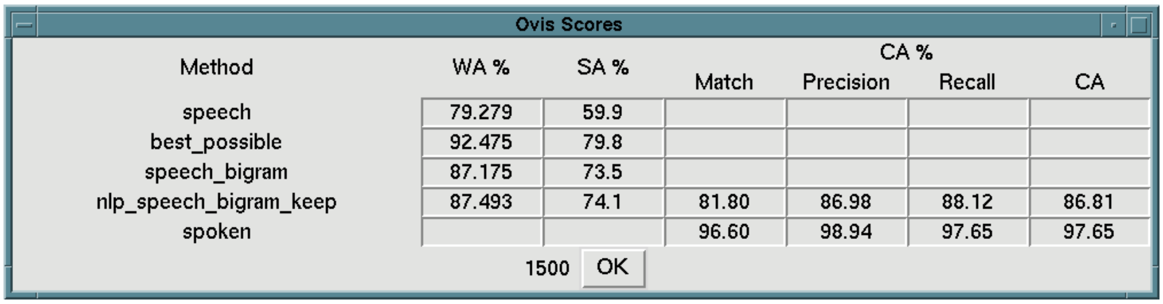

Figure 6: This figure shows the proportion of inputs (word graphs in this case) (percentage of the test-set of 380) can be treated per amount of CPU-time (in milliseconds) for a number of different parsers (a head-corner parser *hc*, a left-corner parser *lc*, an *inactive* chart parser, an *active* chart parser, a bottom-up Earley parser *bu-earley* and an LR parser *lr_cyk*. Note that in this example the parsers parse all paths through the word graph. For this particular test-set the head-corner parser performs best. As can be seen in the graph it treats 96% of the input word-graphs within 200 milliseconds.

times and memory usage for each of those parsers. For example, figure 5 is the result of a test run of 5000 word graphs for four different parsers. For slower parsers it is useful to implement a time-out to make sure that test sets can be treated within a reasonable amount of time. In such cases mean cputime does not make sense; therefore, it is also possible to obtain a graph in which the percentage of inputs that can be completed within a certain amount of cputime is displayed. This is supported in Hdrug as well; an example is given in figure 6. Similar support is provided for the analysis of a given test-set of sentences with respect to input size and with respect to the number of readings assigned.

The functionality of Hdrug has been extended in various ways for the OVIS application. For example, a procedure has been implemented which can be used to generate random sentences, as a means to find errors in the grammar. The menu bar is extended with a new menu-button which provides an interface to this new feature. Incorporating such new features in the user interface is very straightforward.

Furthermore, similar to the View menu of Ale it is also possible to obtain visualisation of datastructures such as lexical entries and grammar rules. This menu also provides an interface for the visualisation of word graphs by piping these word graphs to either the VCG (Sander, 1995) or *dotty* (Koutsofios and North, 1994) graph drawing tools.

Apart from adding new menu buttons it is also easy to add items to existing pull-down menus. For example, in OVIS we are not only interested in the speed of the parser, but also in the accuracy. A component has been implemented which measures word accuracy, sentence accuracy and concept accuracy (by comparing the results of analysis with a given annotation). This functionality is available through a number of new items on the Test-Suite menu. If a test suite has been loaded, then we can use this component to measure word accuracy and sentence accuracy of a number of difference analysis methods. Information is displayed in a window which is updated every now and then (the interval can be set by the user). Such an information window looks as in figure 7.



Ovis Scores

| Method | WA % | SA % | CA % | | | |
|---|---|---|---|---|---|---|
| | | | Match | Precision | Recall | CA |
| speech | 79.279 | 59.9 | | | | |
| best_possible | 92.475 | 79.8 | | | | |
| speech_bigram | 87.175 | 73.5 | | | | |
| nlp_speech_bigram_keep | 87.493 | 74.1 | 81.80 | 86.98 | 88.12 | 86.81 |
| spoken | | | 96.60 | 98.94 | 97.65 | 97.65 |

1500 OK

Figure 7: Example of an extension to Hdrug as part of the Ovis development system. Such extensions can be defined by means of a TCL script. The integration of such extensions with the Hdrug user interface is trivial.

## 5 Final remarks

The main characteristics of Hdrug are its extendability and flexibility. We believe that if such systems are useful for computational linguists, then these two criteria are of extreme importance.

## Acknowledgments

Part of this research is being carried out within the framework of the Priority Programme Language and Speech Technology (TST). The TST-Programme is sponsored by NWO (Dutch Organisation for Scientific Research).

## References

Gosse Bouma and Gertjan van Noord. 1993. Head-driven parsing for lexicalist grammars: Experimental results. In *Sixth Conference of the European Chapter of the Association for Computational Linguistics*, Utrecht.

Gosse Bouma and Gertjan van Noord. 1994. Constraint-based categorial grammar. In *32th Annual Meeting of the Association for Computational Linguistics*, New Mexico.

Lou Boves, Jan Landsbergen, Remko Scha, and Gertjan van Noord. 1995. *Language and Speech Technology.* NWO Den Haag. Project plan for the NWO Priority Programme 'Language and Speech Technology'.

Bob Carpenter. 1992. The attribute logic engine user guide. Technical report, Laboratory for Computatonal Linguistics, Carnegie Mellon University, Pittsburgh.

Karten Konrad, Holger Maier, and Manfred Pinkal. 1996. CLEARS — an education and research tool for computational semantics. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Copenhagen.

Eleftherios Koutsofios and Stephen C. North. 1994. Editing graphs with *dotty*. *dotty* User Manual.

C.S. Mellish. 1988. Implementing systemic classification by unification. *Computational Linguistics*, 14(1).

Richard A. O'Keefe. 1990. *The Craft of Prolog.* The MIT Press.

Fernando C.N. Pereira and David Warren. 1980. Definite clause grammars for language analysis — a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13.

Fernando C.N. Pereira. 1981. Extraposition grammars. *Computational Linguistics*, 7(4).

Peter Ross. 1989. *Advanced Prolog.* Addison-Wesley.

G. Sander. 1995. Graph layout through the VCG tool. In R. Tamassia and I.G. Tollis, editors, *Graph Drawing, DIMACS International Workshop GD '94, Proceedings; Lecture Notes in Computer Science 894*, pages 194–205. Springer Verlag.

Stuart M. Shieber, Gertjan van Noord, Robert C. Moore, and Fernando C.N. Pereira. 1989. A semantic-head-driven generation algorithm for unification based formalisms. In *27th Annual Meeting of the Association for Computational Linguistics*, pages 7–17, Vancouver.

Gertjan van Noord and Gosse Bouma. 1994. Adjuncts and the processing of lexical rules. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, Kyoto.

Gertjan van Noord, Gosse Bouma, Rob Koeling, and Mark-Jan Nederhof. 1996. Conventional natural language processing in the NWO priority programme on language and speech technology. October 1996 Deliverables. Technical Report 28, NWO Priority Programme Language and Speech Technology.

Gertjan van Noord. 1994. Head corner parsing for TAG. *Computational Intelligence*, 10(4).