Unification-Based Grammar Engineering

~~Dan Flickinger~~
~~Stanford University~~ & Redbird Advanced Learning
danf@stanford.edu

Stephan Oepen
Oslo University
oe@ifi.uio.no

ESSLLI 2016; August 15–19, 2016

Outlook: The English Resource Grammar (ERG)

Development Background (1993 – today)

- General-purpose, wide-coverage, computational English grammar;
- mainly Dan Flickinger, with Malouf, Bender, Smith, Oepen, others;
- supported in multiple HPSG processing environments (LKB & PET);
- coverage of 85 – 98 % of running text across genres and domains;
- multitude of research users and applications; a few companies.

http://erg.delph-in.net/

DELPH-IN: Deep Linguistic Processing with HPSG

Inbox (59,329) - ste... | Google Calendar - Week ... | Unification-Based Gram... | RecentChanges - De... | English Resource Gramm... | Home - DELPH-IN

www.delph-in.net/ latex strikethrough

DELPH-IN SEARCH

Some History (PDF)

- Home
- Background
- Software
- Grammars
- Treebanks
- News & Events
- Links
- Contact

Home

Computational linguists from research sites world-wide have joined forces in a collaborative effort aimed at 'deep' linguistic processing of human language. The goal is the combination of linguistic and statistical processing methods for getting at the meaning of texts and utterances. The partners have adopted Head-Driven Phrase Structure Grammar (HPSG) and Minimal Recursion Semantics (MRS), two advanced models of formal linguistic analysis. They have also committed themselves to a shared format for grammatical representation and to a rigid scheme of evaluation, as well as to the general use of open-source licensing and transparency.

The collaboration involves researchers from the following institutions:

- Bulgarian Academy of Science (Bulgaria), Linguistic Modeling Department
- Cambridge University (UK), Computer Laboratory
- DFKI Saarbrücken GmbH (Germany), Language Technology Lab (co-founder)
- Kyung Hee University (Korea), School of English
- Melbourne University (Australia), Language Technology Group
- NTT Communication Science Laboratories (Japan), Linguistic Intelligence Research Group
- Nanyang Technological University (Singapore), Division of Linguistics and Multilingual Studies
- Norwegian University of Science and Technology (Norway), Department of Language and Communication Studies
- Saarland University (Germany), Department for Computational Linguistics

Home | Background | Software | Grammars | Treebanks | News & Events | Links | Contact

The DELPH-IN infrastucture is hosted at the University of Oslo

The 2016 Certified Grammar Engineer(s) Contest

Success Criteria

No load-time errors; additional test items; full coverage, no over-generation, no spurious ambiguity; one fact, one place; documentation on thought process; capitalization, indentation, and whitespace; signs of emacs(1) use.

The 2016 Certified Grammar Engineer(s) Contest

Success Criteria

No load-time errors; additional test items; full coverage, no over-generation, no spurious ambiguity; one fact, one place; documentation on thought process; capitalization, indentation, and whitespace; signs of emacs(1) use.

ESSLI339 (Thursday, 18:01)

- Fairly rudimentary solution; really just getting started on Exercise 2.

The 2016 Certified Grammar Engineer(s) Contest

Success Criteria

No load-time errors; additional test items; full coverage, no over-generation, no spurious ambiguity; one fact, one place; documentation on thought process; capitalization, indentation, and whitespace; signs of emacs(1) use.

ESSLI339 (Thursday, 18:01)

- Fairly rudimentary solution; really just getting started on Exercise 2.

ESSLI146 (Thursday, 23:51)

- Near-perfect solution to Exercise 3; nice solution to modifier ordering;
- remaining under-generation: sentence-initial adverbs blocked in rule.

The 2016 Certified Grammar Engineer(s) Contest

Success Criteria

No load-time errors; additional test items; full coverage, no over-generation, no spurious ambiguity; one fact, one place; documentation on thought process; capitalization, indentation, and whitespace; signs of emacs(1) use.

ESSLI339 (Thursday, 18:01)

- Fairly rudimentary solution; really just getting started on Exercise 2.

ESSLI146 (Thursday, 23:51)

- Near-perfect solution to Exercise 3; nice solution to modifier ordering;
- remaining under-generation: sentence-initial adverbs blocked in rule.

ESSLI311 (Friday, 08:49)

- Functionally perfect solution to Exercise 4; tiny redundancy residues.

Summing up Basic Terminology

Government — Agreement — Licensing

*The dog barks. — *The dog a cat barks — *The dog barks a cat.*
*Kim depends on Sandy — *Kim depends in Sandy*
The class meets everyday in room 5.18 at 17:00.

- **Constituent** node in analysis tree (terminal or instantiation of rule);
- **Government** directed: a property of c_1 determines the form of c_2;
- **Agreement** bi-directional: co-occurence of properties on c_1 and c_2.
- **Head** licenses additional constituents and can govern their form;
- **Specifier** precedes head, singleton, nominative case, agreement;
- **Complement** post-head, licensed and governed, order constraints;
- **Adjunct** 'free' modifier, optional, may iterate, designated position.

Composition: Appending Lists with Unification

- A *difference list* embeds an open-ended list into a container structure that provides a 'pointer' to the end of the ordinary list at the top level:

$$\boxed{A}\ \textit{*dlist*}\begin{bmatrix}\texttt{LIST}\ \boxed{1}\ \textit{*ne-list*}\begin{bmatrix}\texttt{FIRST}\ X\\ \texttt{REST}\ \ \boxed{2}\ \textit{*list*}\end{bmatrix}\\ \texttt{LAST}\ \boxed{2}\end{bmatrix}\qquad \boxed{B}\ \textit{*dlist*}\begin{bmatrix}\texttt{LIST}\ \boxed{3}\ \textit{*ne-list*}\begin{bmatrix}\texttt{FIRST}\ Y\\ \texttt{REST}\ \ \boxed{4}\ \textit{*list*}\end{bmatrix}\\ \texttt{LAST}\ \boxed{4}\end{bmatrix}$$

- Using the `LAST` pointer of difference list $\boxed{A}$ we can append $\boxed{A}$ and $\boxed{B}$ by

  (i) unifying the front of $\boxed{B}$ (i.e. the value of its `LIST` feature) into the tail of $\boxed{A}$ (i.e. the value of its `LAST` feature); and

  (ii) using the tail of $\boxed{B}$ as the new tail for the result of the concatenation.

An Example: Concatenation of Orthography

$$\begin{bmatrix} \texttt{ORTH} \begin{bmatrix} \texttt{LIST} & \boxed{1} \\ \texttt{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix} \longrightarrow \begin{bmatrix} \texttt{ORTH} \begin{bmatrix} \texttt{LIST} & \boxed{1} \\ \texttt{LAST} & \boxed{2} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \texttt{ORTH} \begin{bmatrix} \texttt{LIST} & \boxed{2} \\ \texttt{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix}$$

An Example: Concatenation of Orthography

$$\begin{bmatrix} \text{ORTH} \begin{bmatrix} \text{LIST} & \boxed{1} \\ \text{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix} \longrightarrow \begin{bmatrix} \text{ORTH} \begin{bmatrix} \text{LIST} & \boxed{1} \\ \text{LAST} & \boxed{2} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} \text{ORTH} \begin{bmatrix} \text{LIST} & \boxed{2} \\ \text{LAST} & \boxed{3} \end{bmatrix} \end{bmatrix}$$

```
binary-rule := phrase &
[ ORTH [ LIST #front, LAST #tail ],
  ARGS < [ ORTH [ LIST #front, LAST #middle ] ],
         [ ORTH [ LIST #middle, LAST #tail ] ] > ].

binary-head-initial := head-initial & binary-rule.

binary-head-final := head-final & binary-rule.
```

Notational Conventions

- lists not available as built-in data type; abbreviatory notation in TDL:

  `< a, b >` ≡ `[ FIRST a, REST [ FIRST b, REST *null* ] ]`

- underspecified (variable-length) list:

  `< a ... >` ≡ `[ FIRST a, REST *list* ]`

- difference (open-ended) lists; allow concatenation by unification:

  `<! a !>` ≡ `[ LIST [ FIRST a, REST #tail ], LAST #tail ]`

- built-in and ‘non-linguistic’ types pre- and suffixed by asterisk (**top**);
- strings (e.g. *“chased”*) need no declaration; always subtypes of **string**;
- strings cannot have subtypes and are (thus) mutually incompatible.

Our Grammars: Table of Contents

Type Description Language (TDL)

- `types.tdl` type definitions: hierarchy of grammatical knowledge;
- `lexicon.tdl` instances of (lexical) types plus orthography;
- `rules.tdl` instances of construction types; used by the parser;
- `lrules.tdl` lexical rules, applied before non-lexical rules;
- `irules.tdl` lexical rules that require orthographemic variation;
- `roots.tdl` grammar start symbol(s): ‘selection’ of final results.

Auxiliary Files (Grammar Configuration for LKB)

- `labels.tdl` TFS templates abbreviating node labels in trees;
- `globals.lsp`, `user-fns.lsp` parameters and interface functions.