Overview of goals and methods

Outline

1. Overview of goals and methods
2. Implementation platform and formalism
3. Treebanks and output formats
4. Semantic phenomena
5. Parameter tuning for applications
6. System enhancements underway
7. Sample applications using ERS
Overview of goals and methods

What is an ERS?

- A rich, spanning, compositionally produced representation of sentence meaning, including ‘who did what to whom’, grammatically required coreference, and grammatically constrained scope information.
- *Precision semantic dependencies are useful and readily available.*
What can I get from an ERS?

- High-precision semantic relations, including long-distance dependencies
- (Partial) information about the scope of scopal operators
- Information about tense, number and similar features
Where does an ERS come from?

- The English Resource Grammar: A hand-crafted, broad-coverage, open source, HPSG grammar for English
  [Flickinger, 2000, Flickinger, 2011]
- Developed over 23 years, against text from varied genres:
  - Meeting scheduling dialogues, tourism brochures, customer email, Wikipedia articles on compiling, newspaper text (WSJ), online forum posts, & more
  - But not genre- or domain-dependent.
- Efficient parsing algorithms + maxent parse selection, trained on grammar-derived treebanks
  [Callmeier, 2002, Oepen et al., 2004, Toutanova et al., 2005]
How can I get ERS?

- ERG-based parsing
  - With PET (included in the LOGON distribution) [Callmeier, 2002]
  - With ACE [Callmeier, 2002]
- Interactive single-sentence and batch parsing
- Software components with APIs for inclusion in NLP systems
How can I get ERS?

ERS-annotated sembanks (manually-verified analyses)

<table>
<thead>
<tr>
<th>Treebank</th>
<th>Sents</th>
<th>Words</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepBank</td>
<td>36918</td>
<td>760K</td>
<td>Wall Street Journal newspaper text (as in PTB)</td>
</tr>
<tr>
<td>LOGON</td>
<td>11559</td>
<td>160K</td>
<td>Tourism brochures</td>
</tr>
<tr>
<td>Verbmobil</td>
<td>11407</td>
<td>90K</td>
<td>Transcribed dialogues</td>
</tr>
<tr>
<td>WeScience</td>
<td>9265</td>
<td>170K</td>
<td>100 articles on Comp Ling in Wikipedia</td>
</tr>
<tr>
<td>SemCor</td>
<td>2634</td>
<td>50K</td>
<td>Subset of Brown corpus, word-sense-annotated</td>
</tr>
<tr>
<td>E-commerce</td>
<td>5413</td>
<td>50K</td>
<td>Customer emails</td>
</tr>
<tr>
<td>Misc</td>
<td>3423</td>
<td>40K</td>
<td>Test suites, essay, online forum</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>80619</td>
<td>1320K</td>
<td></td>
</tr>
</tbody>
</table>

http://www.delph-in.net/redwoods
Overview of goals and methods

How can I get ERS?

In a wide variety of formats:

- **MRS** [Copestake, 2002, Copestake et al., 2005] Underspecified logical form with variables
- **Dependency MRS (DMRS)** [Copestake, 2009] Variable-free semantic dependency graph including scope
- **Elementary Dependency Structures (EDS)** [Oepen and Lønning, 2006] Variable-free semantic dependency graph without scope
- **Bilexical semantic dependencies (DM)** [Ivanova et al., 2012] Only word-to-word dependencies

For this tutorial, we will mostly use ‘standard’ MRS, and sometimes DMRS
Goals of this tutorial

- Set up the ERG-based parsing stack, including preprocessing
- Access ERG Redwoods/DeepBank treebanks in the various export formats
- Interpret ERS representations
Outline

1. Overview of goals and methods
2. Implementation platform and formalism
3. Treebanks and output formats
4. Semantic phenomena
5. Parameter tuning for applications
6. System enhancements underway
7. Sample applications using ERS
Installation of parser and grammar

- Install VirtualBox from VirtualBox.org
- Download the Ubuntu+ERS appliance file from UW
- Run VirtualBox, and from File menu, choose “Import Appliance”
- Choose the ERS appliance file to start the import wizard
- When finished with the wizard, start the new virtual machine
Contents of the package

- ACE parser/generator
- English Resource Grammar (ERG)
- Linguistic User Interface (LUI)
- Full-Forest Treebanker
Running the parser interactively

- In a terminal window in VirtualBox, start the parser:
  `ace -g erg/erg.dat -1lTf`
- Type a simple test sentence, and hit Enter:
  \textit{Most fierce dogs chase cats.}
- A separate parse tree window pops up.
- Right-click within the parse tree window, and choose “Indexed MRS” to see a compressed view of the ERS.

Alternatively, to get the ERS as a string written to the terminal:

- In the terminal window, start the parser without LUI:
  `ace -g erg/erg.dat -1Tf`
- Type a sentence, and hit Enter:
  \textit{Most fierce dogs chase cats.}
- The ‘native’ or ‘simple’ ERS output appears in the terminal window.
Running the parser in batch mode

- Create a file “mysents.txt” containing a small set of sentences, with one sentence per line
- Run the parser with this filename as an additional argument, and store the results in a file called “myoutput.txt”
  `ace -g erg.dat -lT mysents.txt > myoutput.txt`
- Open the file “myoutput.txt” to see the results of the batch parsing
Example sentence 1

*Most house cats are easy for dogs to chase.*

\[
\langle h_1, e_3, \\
  h_4 : \_most\_q(x_5, h_6, h_7), \\
  h_8 : \text{compound}(e_{10}, x_5, x_9), \\
  h_{11} : \text{undef\_q}(x_9, h_{12}, h_{13}), \\
  h_{14} : \_house\_n\_of(x_9, i_{15}), \\
  h_8 : \_cat\_n\_1(x_5), \\
  h_2 : \_easy\_a\_for(e_3, h_{16}, x_{17}), \\
  h_{18} : \text{undef\_q}(x_{17}, h_{19}, h_{20}), \\
  h_{21} : \_dog\_n\_1(x_{17}), \\
  h_{22} : \_chase\_v\_1(e_{23}, x_{17}, x_5) \\
\{ h_1 = qh_2, h_6 = qh_8, h_{12} = qh_{14}, h_{16} = qh_{22}, h_{19} = qh_{21} \}\rangle
\]
Example sentence 2

"Which book did the guy who left give to his neighbor?"

\[ \langle h_1, e_3, \]
\[ h_4: \_ \text{which}_q(x_5, h_6, h_7), \]
\[ h_8: \_ \text{book}_n \_ \text{of}(x_5, i_9), \]
\[ h_{10}: \_ \text{the}_q(x_{12}, h_{13}, h_{11}), \]
\[ h_{14}: \_ \text{guy}_n \_ \text{1}(x_{12}), \]
\[ h_{14}: \_ \text{leave}_v \_ \text{1}(e_{15}, x_{12}, i_{16}), \]
\[ h_2: \_ \text{give}_v \_ \text{1}(e_3, x_{12}, x_5, x_{17}), \]
\[ h_{18}: \text{def}_n \_ \text{explicit}_q(x_{17}, h_{20}, h_{19}), \]
\[ h_{21}: \text{poss}(e_{23}, x_{17}, x_{22}), \]
\[ h_{24}: \text{pronoun}_q(x_{22}, h_{25}, h_{26}), \]
\[ h_{27}: \text{pron}(x_{22}), \]
\[ h_{21}: \_ \text{neighbor}_n \_ \text{1}(x_{17}) \]
\[ \{ h_1 = q h_2, h_6 = q h_8, h_{13} = q h_{14}, h_{20} = q h_{21}, h_{25} = q h_{27} \} \]
Disambiguation alternatives

- Automatic one-best, using maxent model:
  Have the parser only produce the one most likely analysis for each input.

- Manual selection, using ACE Treebanker:
  Have the parser produce all analyses, with the forest presented via discriminants which enable manual selection of the intended analysis.
Introduction to ERS formalism

The cat sleeps.

\[ \langle h_1, e_3, \begin{align*}
h_4 &: \text{the}_q(x_6, h_7, h_5), \\
h_8 &: \text{cat}_n_1(x_6), \\
h_2 &: \text{sleep}_v_1(e_3, x_6) \\
\{ h_1 &= q h_2, h_7 = q h_8 \} \rangle \]

- Top handle
- Bag of elementary predications
- Scope constraints
ERS variable types

\[
\begin{align*}
\text{u (underspecified)} & \\
\text{i (individual)} & \\
\text{e (eventuality)} & \quad \quad \quad \text{x (instance)} & \quad \quad \quad \text{h (handle)} & \text{p}
\end{align*}
\]
Properties of variables

- Number, person, gender, and individuation on instances

\[ h_8 : \text{\texttt{cat\_n\_1}}(\text{ARG0 } x_6 \{\text{PERS 3, NUM sg, GEND n, IND +}\}) \]

- Sentence force, tense, mood, and aspect on eventualities

\[ h_2 : \text{\texttt{sleep\_v\_1}}( \text{ARG0 } e_3 \{\text{SF prop, TENSE pres, MOOD \textit{indicative}, PROG -, PERF -}\}, \text{ARG1 } x_6) \]
Every predication contains

- Predicate name
- Label of type \textit{handle}
- Intrinsic argument of type \textit{individual} as value of ARG0 (except that the ARG0 of quantifiers is not intrinsic)

Predications may contain additional arguments, as values of attributes normally called ARG1, ARG2, ..., though quantifiers and conjunctions, among others, use a richer inventory of attribute names.
Predicates

Surface vs. abstract:

- Naming conventions for **surface** predicates (from lexical entries)
  - Leading underscore
  - Underscore-separated fields
    `_lemma_pos_sense`
    - *lemma* is orthography of the base form of word in lexicon
    - *pos* draws coarse-grained sense distinction
    - *sense* draws finer-grained sense distinction

- **Abstract** predicates are introduced either via construction, or in decomposed semantics of lexical entries.
Abstract predicate example: Noun-noun compounds

The police dog barked.

\[ \langle \text{"The police dog barked."} \rangle = \langle \text{"} h_8: \text{compound}(e_{10}, x_6, x_9), h_{14}: _\text{police}\_n\_1(x_9), h_8: _\text{dog}\_n\_1(x_6) \langle \rangle \rangle \]
Words for named entities introduce in their semantic predication a parameter as the value of a distinguished attribute CARG

We admire Kim greatly.

\( h_{13} : \text{named}(x_9, \text{Kim}) \)
A predication may have a handle as the value of one of its argument attributes, with a corresponding element in the HCONS list identifying the label of the highest-scoping predication of the argument phrase.

We know that the cat didn’t sleep.

\[
\langle h_1, e_3, \\
h_4: \text{pron}(x_5),
h_6: \text{pronoun}_q(x_5, h_7, h_8),
h_2: \text{know}_v_1(e_3, x_5, h_9),
h_{10}: \text{the}_q(x_{12}, h_{13}, h_{11}),
h_{14}: \text{cat}_n_1(x_{12}),
h_{15}: \text{neg}(e_{17}, h_{16}),
h_{18}: \text{sleep}_v_1(e_{19}, x_{12}) \\
\{ h_1 = q h_2, h_7 = q h_4, h_9 = q h_{15}, h_{13} = q h_{14}, h_{16} = q h_{18} \} \rangle
\]
Basic assumptions for well-formed ERS

- Every predication has a unique ‘intrinsic’ ARG0 (not quantifiers)
- Every instance variable is bound by a quantifier
- Scope resolution results in a set of one or more trees (which can be treated as conventional logical forms)
Outline

1. Overview of goals and methods
2. Implementation platform and formalism
3. Treebanks and output formats
4. Semantic phenomena
5. Parameter tuning for applications
6. System enhancements underway
7. Sample applications using ERS
Introduction to the treebanks

Several collections of text in a variety of domains
80,000 sentences, 1.3 million words
Each sentence parsed with ERG to produce candidate analyses
Manually disambiguated via syntactic or semantic *discriminants* [Carter, 1997, Oepen et al., 2004]
Each correct analysis stored with its semantic representation
Semantic search via fingerprints

- Identify elements of ERS to match in treebank
- Regular expressions over predicate names
- Returns sentences and their ERS (in multiple views)
- Useful for exploring ERS in support of feature design
Fingerprint search example: ‘Object’ Control

Semantic Search Interface - DeepBank

Query

```
[ARG2 x, ARG3 h1]
h2:*_v_*[ARG0 e, ARG1 x]
{ h1 = q h2 }
```

Format  MRS  EDS  DM  Results per Page 20  search

Show SPARQL

Results Page 1

20004005 Longer maturities are thought to indicate declining interest rates because they permit portfolio managers to retain relatively higher rates for a longer period.
Fingerprint formalism

- Partial descriptions of ERSs automatically expanded to SPARQL queries for efficient search over RDF encoding of the sembank [Kouylekov and Oepen, 2014].
- Queries consist of one or more EP descriptions, separated by white space, plus optionally HCONS lists.
- EP descriptions consist of one or more of:
  - Identifier (label, e.g. h0)
  - (Lucene-style pattern over) predicate symbol (e.g. *v*)
  - List of argument roles with (typed) value identifiers (e.g. [ARG1 x2])
- Repeated identifiers across EPs indicate required reentrancies in the matched ERSs
For more information

- Fuller description of query language:
  http://sdp.delph-in.net/2015/search.html

- Sample fingerprints in ERG Semantic Documentation phenomenon pages
  http://moin.delph-in.net/ErgSemantics

- Further examples later in this tutorial
Available output formats

- Standard MRS
- Simple MRS
- DMRS
- EDS
- Bi-lexical dependencies
- Direct ERS output from ACE
Standard MRS (terse)

The jungle lion was chasing a small giraffe.

\[
\langle h_1, e_3, \quad h_4: \text{the}_q(x_6, h_7, h_5), \quad h_8: \text{compound}(e_{10}, x_6, x_9), \quad h_{11}: \text{udef}_q(x_9, h_{12}, h_{13}), \quad h_{14}: \text{jungle}_n_1(x_9), \quad h_8: \text{lion}_n_1(x_6), \\
\quad h_2: \text{chase}_v_1(e_3, x_6, x_{15}), \quad h_{16}: \text{a}_q(x_{15}, h_{18}, h_{17}), \quad h_{19}: \text{small}_a_1(e_{20}, x_{15}), \quad h_{19}: \text{giraffe}_n_1(x_{15}) \quad \{ h_1 = q h_2, h_7 = q h_8, h_{12} = q h_{14}, h_{18} = q h_{19} \} \rangle
\]
The jungle lion was chasing a small giraffe.

\[
\langle h_1, e_3, \\
\begin{align*}
h_4 & : \text{the}_q(\text{ARG0 } x_6, \text{RSTR } h_7, \text{BODY } h_5), \\
h_8 & : \text{compound}(\text{ARG0 } e_{10}, \text{ARG1 } x_6, \text{ARG2 } x_9), \\
h_{11} & : \text{udef}_q(\text{ARG0 } x_9, \text{RSTR } h_{12}, \text{BODY } h_{13}), \\
h_{14} & : \_\text{jungle}_n \_1(\text{ARG0 } x_9), \\
h_8 & : \_\text{lion}_n \_1(\text{ARG0 } x_6), \\
h_2 & : \_\text{chase}_v \_1(\text{ARG0 } e_3, \text{ARG1 } x_6, \text{ARG2 } x_{15}), \\
h_{16} & : \_\text{a}_q(\text{ARG0 } x_{15}, \text{RSTR } h_{18}, \text{BODY } h_{17}), \\
h_{19} & : \_\text{small}_a _1(\text{ARG0 } e_{20}, \text{ARG1 } x_{15}), \\
h_{19} & : \_\text{giraffe}_n _1(\text{ARG0 } x_{15})
\end{align*}
\}
\]
Standard MRS with argument roles and properties

The jungle lion was chasing a small giraffe.

\[\langle h_1,e_3,\]
\[h_4 : \text{the}_q(\text{ARG0 } x_6,\]
\[\quad\text{RSTR } h_7, \text{BODY } h_5)\]
\[h_8 : \text{compound}(\text{ARG0 } e_{10}\{\text{SF prop, TENSE untensed, MOOD indic, PROG -}, \text{PERF -}\},\]
\[\quad\text{ARG1 } x_6, \text{ARG2 } x_9\{\text{IND +}\})\]
\[h_{11} : \text{undefined}_q(\text{ARG0 } x_9, \text{RSTR } h_{12}, \text{BODY } h_{13}),\]
\[h_{14} : \text{jungle}_n_1(\text{ARG0 } x_9),\]
\[h_8 : \text{lion}_n_1(\text{ARG0 } x_6\{\text{PERS 3, NUM sg, IND +}\}),\]
\[h_2 : \text{chase}_v_1(\text{ARG0 } e_3\{\text{SF prop, TENSE past, MOOD indic, PROG +}, \text{PERF -}\},\]
\[\quad\text{ARG1 } x_6, \text{ARG2 } x_{15}\{\text{PERS 3, NUM sg, IND +}\})\]
\[h_{16} : \text{a}_q(\text{ARG0 } x_{15}, \text{RSTR } h_{18}, \text{BODY } h_{17}),\]
\[h_{19} : \text{small}_a_1(\text{ARG0 } e_{20}\{\text{SF prop, TENSE untensed, MOOD indic}\}, \text{ARG1 } x_{15}),\]
\[h_{19} : \text{giraffe}_n_1(\text{ARG0 } x_{15})\]
\{ h_1 = q h_2, h_7 = q h_8, h_{12} = q h_{14}, h_{18} = q h_{19} \} \]
Standard MRS also with character positions

The jungle lion was chasing a small giraffe.

\[
\langle \ h_1, e_3, \\
\quad h_4 : \_ \text{the}_q \langle 0 : 3 \rangle (\text{ARG0} \ x_6, \text{RSTR} \ h_7, \text{BODY} \ h_5), \\
\quad h_8 : \text{compound} \langle 4 : 15 \rangle (\text{ARG0} \ e_{10}, \text{ARG1} \ x_6, \text{ARG2} \ x_9), \\
\quad h_{11} : \text{undef}_q \langle 4 : 10 \rangle (\text{ARG0} \ x_9, \text{RSTR} \ h_{12}, \text{BODY} \ h_{13}), \\
\quad h_{14} : \_ \text{jungle}_n_1 \langle 4 : 10 \rangle (\text{ARG0} \ x_9), \\
\quad h_8 : \_ \text{lion}_n_1 \langle 11 : 15 \rangle (\text{ARG0} \ x_6), \\
\quad h_2 : \_ \text{chase}_v_1 \langle 20 : 27 \rangle (\text{ARG0} \ e_3, \text{ARG1} \ x_6, \text{ARG2} \ x_{15}), \\
\quad h_{16} : \_ \text{a}_q \langle 28 : 29 \rangle (\text{ARG0} \ x_{15}, \text{RSTR} \ h_{18}, \text{BODY} \ h_{17}), \\
\quad h_{19} : \_ \text{small}_a_1 \langle 30 : 35 \rangle (\text{ARG0} \ e_{20}, \text{ARG1} \ x_{15}), \\
\quad h_{19} : \_ \text{giraffe}_n_1 \langle 36 : 44 \rangle (\text{ARG0} \ x_{15}) \\
\{ h_1 = q \ h_2, \ h_7 = q \ h_8, \ h_{12} = q \ h_{14}, \ h_{18} = q \ h_{19} \ \}\n\]
The jungle lion was chasing a small giraffe.
The jungle lion was chasing a small giraffe.
EDS: Elementary Dependency Structures

- Reduction to core predicate-argument graph [Oepen et al., 2002];
- ‘semantic network’: formally (if not linguistically) similar to AMR.

The jungle lion was chasing a small giraffe.

(e3 / _chase_v_1
  :ARG1 (x6 / _lion_n_1
    :ARG1-of (e10 / compound
      :ARG2 (x9 / _jungle_n_1
        :BV-of (_2 / udef_q)))
    :BV-of (_1 / _the_q)))
  :ARG2 (x15 / _giraffe_n_1
    :ARG1-of (e20 / _small_a_1
      :BV-of (_3 / _a_q)))
DM: *Bi-Lexical* Semantic Dependencies

- Lossy reduction of EDS graph: use only surface tokens as nodes;
- construction semantics as edge labels; coarse argument frames;

→ Oepen et al. on Friday: *Comparability of Linguistic Graph Banks*.

```
The jungle lion was chasing a small giraffe.
```

![Diagram of semantic dependencies with labeled nodes and edges]
ERS output directly from ACE parser

The jungle lion was chasing a small giraffe.

[ LTOP: h0
INDEX: e2 [ e SF: prop TENSE: past MOOD: indicative PROG: + PERF: - ]
RELS: < [ _the_q_rel<0:3> LBL: h4 ARG0: x3 [ x PERS: 3 NUM: sg IND: + ] RSTR: h5 BODY: h6 ]
[ udef_q_rel<4:10> LBL: h10 ARG0: x9 RSTR: h11 BODY: h12 ]
[ "_jungle_n_1_rel"<4:10> LBL: h13 ARG0: x9 ]
[ "_lion_n_1_rel"<11:15> LBL: h7 ARG0: x3 ]
[ "_chase_v_1_rel"<20:27> LBL: h1 ARG0: e2 ARG1: x3 ARG2: x14 [ x PERS: 3 NUM: sg IND: + ] ]
[ _a_q_rel<28:29> LBL: h15 ARG0: x14 RSTR: h16 BODY: h17 ]
[ "_small_a_1_rel"<30:35> LBL: h18 ARG0: e19 [ e SF: prop TENSE: untensed MOOD: indicative ] ARG1: x14 ]
[ "_giraffe_n_1_rel"<36:44> LBL: h18 ARG0: x14 ]>
HCONS: < h0 qeq h1 h5 qeq h7 h11 qeq h13 h16 qeq h18 > ]
The jungle lion was chasing a small giraffe.
Inspection and conversion tools

- LUI: inspection
- pyDelphin: conversion and inspection
Interactive disambiguation

Instructions for using ACE Treebanker

- Batch parse a set of sentences
- Invoke the Treebanker with the resulting set of parse forests
- Select a sentence for disambiguation
- Click on each discriminant which is true for the intended analysis
- When the single correct tree remains alone, click “Save”
Outline

1. Overview of goals and methods
2. Implementation platform and formalism
3. Treebanks and output formats
4. Semantic phenomena
5. Parameter tuning for applications
6. System enhancements underway
7. Sample applications using ERS
Sample linguistic analyses

- For individual phenomena, illustrate how they are represented in ERS
- In aggregate, give a sense of the richness of ERS
- Further documentation for many phenomena available at http://moin.delph-in.net/ErgSemantics
Not all surface words are directly reflected in the ERS

It does seem as though Kim will both go and rely on Sandy.

\[
\langle h_1,e_3, \\
\quad h_2: \_\text{seem\_v\_to}(e_3, h_4, i_5), \\
\quad h_6: \_\text{proper\_q}(x_8, h_7, h_9), \\
\quad h_{10}: \_\text{named}(x_8, \text{Kim}), \\
\quad h_{11}: \_\text{go\_v\_1}(e_{12}, x_8), \\
\quad h_{13}: \_\text{and\_c}(e_{14}, h_{11}, e_{12}, h_{15}, e_{16}), \\
\quad h_{15}: \_\text{rely\_v\_on}(e_{16}, x_8, x_{17}), \\
\quad h_{18}: \_\text{proper\_q}(x_{17}, h_{19}, h_{20}), \\
\quad h_{21}: \_\text{named}(x_{17}, \text{Sandy}) \\
\{ \quad h_{19} = q\, h_{21}, \quad h_7 = q\, h_{10}, \quad h_4 = q\, h_{13}, \quad h_1 = q\, h_2 \quad \}\n\]
Sentential negation analyzed in terms of the scopal operator \textit{neg}

\[ \langle h_1, e_3, \\
    h_4: \text{the}_q(x_6, h_7, h_5), \\
    h_8: \text{dog}_n_1(x_6), \\
    h_2: \text{neg}(e_{10}, h_9), \\
    h_{11}: \text{bark}_v_1(e_3, x_6) \\
    \{ h_9 = qh_{11}, h_7 = qh_8, h_1 = qh_2 \} \rangle \]
Contracted negation (*didn’t*, *won’t*) and independent *not* normalized

- The dog did not bark.

\[
\langle h_1, e_3, \\
h_4: _\text{the}_q(x_6, h_7, h_5), \\
h_8: _\text{dog}_n_1(x_6), \\
h_2: \text{neg}(e_{10}, h_9), \\
h_{11}: _\text{bark}_v_1(e_3, x_6) \\
\{ h_9 = q h_{11}, h_7 = q h_8, h_1 = q h_2 \} \rangle
\]
Scope of negation fixed by position in parse tree

Sandy knows that Kim probably didn’t leave.

\[ \langle h_1, e_3, \]
\[ h_4 : \text{proper}_q(x_6, h_5, h_7), \]
\[ h_8 : \text{named}(x_6, \text{Sandy}), \]
\[ h_2 : \text{know}_v_1(e_3, x_6, h_9), \]
\[ h_{10} : \text{proper}_q(x_{12}, h_{11}, h_{13}), \]
\[ h_{14} : \text{named}(x_{12}, \text{Kim}), \]
\[ h_{15} : \text{probable}_a_1(e_{16}, h_{17}), \]
\[ h_{18} : \text{neg}(e_{20}, h_{19}), \]
\[ h_{21} : \text{leave}_v_1(e_{22}, x_{12}, p_{23}) \]
\[ \{ h_{19} = q h_{21}, h_{17} = q h_{18}, h_{11} = q h_{14}, h_9 = q h_{15}, h_5 = q h_8, h_1 = q h_2 \} \]
NP negation treated as generalized quantifier

The body of this quantifier is not fixed by its position in the parse tree

*Kim probably saw no dog.*

$$\langle h_1, e_3, h_4: \text{proper}_q(x_6, h_5, h_7), h_8: \text{named}(x_6, \text{Kim}), h_2: \text{probable}_a_1(e_9, h_{10}), h_{11}: \text{see}_v_1(e_3, x_6, x_{12}), h_{13}: \text{no}_q(x_{12}, h_{15}, h_{14}), h_{16}: \text{dog}_n_1(x_{12}) \rangle$$

\{ $h_{15} = qh_{16}, h_{10} = qh_{11}, h_5 = qh_8, h_1 = qh_2$ \}
Morphological negation unanalyzed (for now)

That dog is invisible.

\[ \langle h_1, e_3, \\
\quad h_4: \text{that}_q \text{dem}(x_6, h_7, h_5), \\
\quad h_8: \text{dog}_n \_1(x_6), \\
\quad h_2: \text{invisible}_a \_\text{to}(e_3, x_6, i_9) \\
\{ h_7 = q h_8, h_1 = q h_2 \} \rangle \]
Lexically negative verbs not decomposed

The dog failed to bark.

\[ \langle h_1, e_3, \]
\[ h_4: \text{the}_q(x_6, h_7, h_5), \]
\[ h_8: \text{dog}_n_1(x_6), \]
\[ h_2: \text{fail}_v_1(e_3, h_9), \]
\[ h_{10}: \text{bark}_v_1(e_{11}, x_6) \]
\[ \{ h_9 = q h_{10}, h_7 = q h_8, h_1 = q h_2 \} \]
Negation interacts with the analysis of sentence fragments

*Not this year.*

\[ h_1, e_3, \]
\[ h_2: \text{unknown}(e_3, u_4), \]
\[ h_2: \text{neg}(e_6, h_5), \]
\[ h_7: \text{loc\_nonsp}(e_8, e_3, x_9), \]
\[ h_{10}: \text{this\_q\_dem}(x_9, h_{12}, h_{11}), \]
\[ h_{13}: \text{year\_n\_1}(x_9) \]
\[ \{ h_{12} = q h_{13}, h_5 = q h_7, h_1 = q h_2 \} \]
Negation fingerprints

\[
\text{neg} [\text{ARG1} \ h1] \\
h2 : [\text{ARG0} \ e] \\
\{ \ h1 = \neg h2 \ }
\]
Some predicates establish required coreference relations

Kim persuaded Sandy to leave.

\[ \langle h_1, e_3, \]
\[ h_4 : \text{proper_q}(x_6, h_5, h_7), \]
\[ h_8 : \text{named}(x_6, Kim), \]
\[ h_2 : \_\text{persuade_v_of}(e_3, x_6, x_{10}, h_9), \]
\[ h_{11} : \text{proper_q}(x_{10}, h_{12}, h_{13}), \]
\[ h_{14} : \text{named}(x_{10}, Sandy), \]
\[ h_{15} : \_\text{leave_v_1}(e_{16}, x_{10}, p_{17}) \]
\[ \{ h_{12} = q h_{14}, h_9 = q h_{15}, h_5 = q h_8, h_1 = q h_2 \} \]
Which arguments are shared is predicate-specific

\[
\langle h_1, e_3, \\
\begin{array}{l}
h_4 : \text{proper\_q}(x_6, h_5, h_7), \\
h_8 : \text{named}(x_6, Kim), \\
h_2 : \text{\_promise\_v\_1}(e_3, x_6, x_{10}, h_9), \\
h_{11} : \text{proper\_q}(x_{10}, h_{12}, h_{13}), \\
h_{14} : \text{named}(x_{10}, Sandy), \\
h_{15} : \text{\_leave\_v\_1}(e_{16}, x_6, p_{17}) \\
\end{array}
\{ h_{12} = q h_{14}, h_9 = q h_{15}, h_5 = q h_8, h_1 = q h_2 \} \rangle
\]

Kim promised Sandy to leave.
Not just verbs can be control predicates

Kim is happy to leave.

\[ \langle h_1, e_3, \]
\[ h_4 : \text{proper}_q(x_6, h_5, h_7), \]
\[ h_8 : \text{named}(x_6, Kim), \]
\[ h_2 : \_\text{happy}_\text{a}_\text{with}(e_3, x_6, h_9), \]
\[ h_{10} : \_\text{leave}_\text{v}_1(e_{11}, x_6, p_{12}) \]
\[ \{ h_9 = q h_{10}, h_5 = q h_8, h_1 = q h_2 \} \]
Control predicates involve diverse syntactic frames; normalized at the semantic level

Kim prevented Sandy from leaving.

\[
\langle h_1, e_3, \\
  h_4 : \text{proper}_q(x_6, h_5, h_7), \\
  h_8 : \text{named}(x_6, \text{Kim}), \\
  h_2 : \text{prevent}_v_{\text{from}}(e_3, x_6, x_{10}, h_9), \\
  h_{11} : \text{proper}_q(x_{10}, h_{12}, h_{13}), \\
  h_{14} : \text{named}(x_{10}, \text{Sandy}), \\
  h_{15} : \text{leave}_v_{1}(e_{16}, x_{10}, p_{17}) \\
  \{ h_{12} = q h_{14}, h_9 = q h_{15}, h_5 = q h_8, h_1 = q h_2 \} \rangle
\]
Control fingerprints

Example: Subject control. [NB: This is a very general search!]

\[
\begin{align*}
&[\text{ARG0 } e1, \text{ ARG1 } x2, \text{ ARG3 } h3] \\
&h4: [\text{ARG0 } e5, \text{ ARG1 } x2] \\
&\{ \text{ h3 } = q \text{ h4 } \}
\end{align*}
\]
Lexically Mediated

Complex examples are easy to find.

\[
\langle h_1, e_3, \quad h_4 : \text{udef}_q(x_6, h_5, h_7), \quad h_8 : \_\text{complex}_a_1(e_9, x_6), \quad h_8 : \_\text{example}_n_\text{of}(x_6, i_{10}), \quad h_2 : \_\text{easy}_a_\text{for}(e_3, h_{11}, i_{12}), \quad h_{13} : \_\text{find}_v_1(e_{14}, i_{12}, x_6) \rangle
\]

\{ h_{11} = q h_{13}, h_5 = q h_8, h_1 = q h_2 \}
Relative clauses

*The cat whose collar you thought I found escaped.*

\[
\langle h_1, e_3, \\
\begin{align*}
&h_4: &\text{the}_q(x_6, h_7, h_5), \\
&h_8: &\text{cat}_n_1(x_6), \\
&h_9: &\text{def\_explicit\_q}(x_{11}, h_{12}, h_{10}), \\
&h_{13}: &\text{poss}(e_{14}, x_{11}, x_6), \\
&h_{15}: &\text{collar}_n_1(x_{11}), \\
&h_{16}: &\text{pron}(x_{17}), \\
&h_{18}: &\text{pronoun}_q(x_{17}, h_{19}, h_{20}), \\
&h_{24}: &\text{think}_v_1(e_{21}, x_{17}, h_{23}, i_{22}), \\
&h_{26}: &\text{pronoun}_q(x_{25}, h_{27}, h_{28}), \\
&h_{29}: &\text{find}_v_1(e_{30}, x_{25}, x_{11}), \\
&h_2: &\text{escape}_v_1(e_{3}, x_{6}, p_{31})
\end{align*}
\text{\{ }h_{27} = q h_{24}, h_{23} = q h_{29}, h_{19} = q h_{16}, h_{12} = q h_{15}, h_7 = q h_8, h_1 = q h_2 \text{ \}}
\]
Right Node Raising

PCBs move into and go out of the machine automatically.

\[ \langle h_1, e_{10}, \\
    h_4 : u\text{def}_q(x_6, h_5, h_7), \\
    h_8 : _\text{pcbs/nns}_u\text{unknown}(x_6), \\
    h_9 : _\text{move}_v_1(e_{10}, x_6), \\
    h_9 : _\text{into}_p(e_{11}, e_{10}, x_{12}), \\
    h_2 : _\text{and}_c(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
    h_{14} : _\text{go}_v_1(e_{13}, x_6), \\
    h_{14} : _\text{out+of}_p\text{dir}(e_{15}, e_{13}, x_{12}), \\
    h_{16} : _\text{the}_q(x_{12}, h_{18}, h_{17}), \\
    h_{19} : _\text{machine}_n_1(x_{12}), \\
    h_2 : _\text{automatic}_a_1(e_{20}, e_3) \rangle \\
\{ h_{18} = q h_{19}, h_5 = q h_8, h_1 = q h_2 \} \]
Long-Distance Dependencies do not constitute a semantic phenomenon

There are no characteristic patterns in the ERS reflecting them

Rather, dependencies which are long-distance in the syntax appear ordinary in the ERS
Outline

1. Overview of goals and methods
2. Implementation platform and formalism
3. Treebanks and output formats
4. Semantic phenomena
5. Parameter tuning for applications
6. System enhancements underway
7. Sample applications using ERS
Parser settings

- Root symbols
- Preprocessing
- Unknown-word handling
- Disambiguation models
- Resource limits
Robust processing: Three methods

- **Csaw:**
  Using probabilistic context-free grammar trained on ERG best-one analyses of 50 million sentences from English Wikipedia
  (Based on previous work on Jigsaw by Yi Zhang)

- **Bridging:**
  Using very general binary bridging constructions added to the ERG which build non-licensed phrases

- **Mal-rules:**
  Using error-specific constructions added to the ERG to admit words or phrases which are predicatably ill-formed, with correct semantics
Efficiency vs. Precision in Parsing

- Parameters to control resource limits
  - **Time**: maximum number of seconds to use per sentence
  - **Memory**: maximum number of bytes to use for building the packed parse forest and for unpacking
  - **Number of analyses**: only unpack part of the forest

- Ubertagging
  Prune the candidate lexical items for each token in a sentence before invoking the parser, using a statistical model trained on Redwoods and DeepBank
  [Dridan, 2013]
Outline

1. Overview of goals and methods
2. Implementation platform and formalism
3. Treebanks and output formats
4. Semantic phenomena
5. Parameter tuning for applications
6. System enhancements underway
7. Sample applications using ERS
Efficiency vs. Precision

- Word senses for finer-grained semantic representations
- More derivational morphology (e.g. semi-productive deverbal nouns)
- Support for coreference within and across sentence boundaries
Information Structure

- Addition of ICONS attribute for constraints on pairs of individuals
- Now used for structurally imposed constraints on *topic* and *focus*
- Passivized subjects (topic) and “topicalized” phrases (focus)
- [Song and Bender, 2012]
Outline

1. Overview of goals and methods
2. Implementation platform and formalism
3. Treebanks and output formats
4. Semantic phenomena
5. Parameter tuning for applications
6. System enhancements underway
7. Sample applications using ERS
Sample applications using ERS

- Scope of negation
- Logic to English (generation)
- Robot blocks world
**Task**

- *SEM2012 Task 1: Identify negation cues and their associated scopes* [Morante and Blanco, 2012]
  - Ex: {The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.
  - Relevant for sentiment analysis, IE, MT, and many other applications
Contribution of ERS

- Operator scope is a first-class notion in ERS
- Scopes discontinuous in the surface string form subgraphs of ERS
- Characterization links facilitate mapping out to string-based annotations
Challenges

- Shared task notions of negation and scope don’t directly match those in ERS
- Target annotations include semantically empty elements
- Dialect differences (early 1900s British English v. contemporary American English)
Approach

- Use cue detection from [Read et al., 2012]
- Map cue identified in string to EP in ERS
- ‘Crawl’ the ERS graph from the cue, according to the type of cue and type of EP encountered
- Use EP characterization and syntactic parse tree to map scope to substrings
- Fall back to [Read et al., 2012] if no parse or top ranked parse has a score of < 0.5
Approach

{The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.

\[
\langle h_1, e_3, \\
\quad h_4: \text{the}_q(x_6, h_7, h_5), \\
\quad h_8: \text{named}(x_6, \text{German}), \\
\quad h_9: \text{send}_v\_\text{for}(e_{10}, i_{11}, x_6), \\
\quad h_9: \text{parg}_d(e_{12}, e_{10}, x_6), \\
\quad h_2: \text{but}_c(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
\quad h_{14}: \text{profess}_v\_\text{to}(e_{13}, x_6, h_{15}), \\
\quad h_{16}: \text{know}_v\_1(e_{17}, x_6, x_{18}), \\
\quad h_{19}: \text{thing}(x_{18}), \\
\quad h_{20}: \text{no}_q(x_{18}, h_{21}, h_{22}), \\
\quad h_{19}: \text{of}_p(e_{23}, x_{18}, x_{24}), \\
\quad h_{25}: \text{the}_q(x_{24}, h_{27}, h_{26}), \\
\quad h_{28}: \text{matter}_n\_\text{of}(x_{24}, i_{29}) \\
\quad \{h_{27} = q h_{28}, h_{21} = q h_{19}, h_{15} = q h_{16}, h_7 = q h_8, h_1 = q h_2\} \rangle
\]
Approach

{The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.

\[
\langle h_1, e_3, \\
 h_4: \text{the}_q(x_6, h_7, h_5), \\
 h_8: \text{named}(x_6, \text{German}), \\
 h_9: \text{send}_v\_v\_for(e_{10}, i_{11}, x_6), \\
 h_9: \text{parg}_d(e_{12}, e_{10}, x_6), \\
 h_2: \text{but}_c(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
 h_{14}: \text{profess}_v\_v\_to(e_{13}, x_6, h_{15}), \\
 h_{16}: \text{know}_v\_1(e_{17}, x_6, x_{18}), \\
 h_{19}: \text{thing}(x_{18}), \\
 h_{20}: \text{no}_q(x_{18}, h_{21}, h_{22}), \\
 h_{19}: \text{of}_p(e_{23}, x_{18}, x_{24}), \\
 h_{25}: \text{the}_q(x_{24}, h_{27}, h_{26}), \\
 h_{28}: \text{matter}_n\_of(x_{24}, i_{29})
\rangle
\{ h_{27} = q h_{28}, h_{21} = q h_{19}, h_{15} = q h_{16}, h_7 = q h_8, h_1 = q h_2 \} \]
Approach

{The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.

⟨h₁,e₃,
  h₄:_the_q(x₆,h₇,h₅),
  h₈:named(x₆,German),
  h₉:_send_v_for(e₁₀,i₁₁,x₆),
  h₉:parg_d(e₁₂,e₁₀,x₆),
  h₂:_but_c(e₃,h₉,e₁₀,h₁₄,e₁₃),
  h₁₄:_profess_v_to(e₁₃,x₆,h₁₅),
  h₁₆:_know_v_1(e₁₇,x₆,x₁₈),
  h₁₉:thing(x₁₈),
  h₂₀:_no_q(x₁₈,h₂₁,h₂₂),
  h₁₉:_of_p(e₂₃,x₁₈,x₂₄),
  h₂₅:_the_q(x₂₄,h₂₇,h₂₆),
  h₂₈:_matter_n_of(x₂₄,i₂₉)
{ h₂₇ = qh₂₈, h₂₁ = qh₁₉, h₁₅ = qh₁₆, h₇ = qh₈, h₁ = qh₂ }⟩
Approach

{The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.

⟨$h_1,e_3,$

$h_4:_\text{the}_q(x_6,h_7,h_5),$
$h_8:_\text{named}(x_6,\text{German}),$
$h_9:_\text{send}_v_\text{for}(e_{10},i_{11},x_6),$
$h_9:_\text{parg}_d(e_{12},e_{10},x_6),$
$h_2:_\text{but}_c(e_3,h_9,e_{10},h_{14},e_{13}),$
$h_{14}:_\text{profess}_v_\text{to}(e_{13},x_6,h_{15}),$
$h_{16}:_\text{know}_v_1(e_{17},x_6,x_{18}),$
$h_{19}:_\text{thing}(x_{18}),$
$h_{20}:_\text{no}_q(x_{18},h_{21},h_{22}),$
$h_{19}:_\text{of}_p(e_{23},x_{18},x_{24}),$
$h_{25}:_\text{the}_q(x_{24},h_{27},h_{26}),$
$h_{28}:_\text{matter}_n_\text{of}(x_{24},i_{29})$

$\{h_{27} = qh_{28}, h_{21} = qh_{19}, h_{15} = qh_{16}, h_7 =qh_8, h_1 = qh_2\}$⟩
Approach

{The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.

\[
\langle h_1,e_3,
\begin{array}{l}
  h_4:\text{the}_q(x_6, h_7, h_5),
  h_8:\text{named}(x_6, \text{German}),
  h_9:\text{send}_v\_\text{for}(e_{10}, i_{11}, x_6),
  h_9:\text{parg}\_d(e_{12}, e_{10}, x_6),
  h_2:\text{but}_c(e_3, h_9, e_{10}, h_{14}, e_{13}),
  h_{14}:\text{profess}_v\_\text{to}(e_{13}, x_6, h_{15}),
  h_{16}:\text{know}_v\_1(e_{17}, x_6, x_{18}),
  h_{19}:\text{thing}(x_{18}),
  h_{20}:\text{no}_q(x_{18}, h_{21}, h_{22}),
  h_{19}:\text{of}_p(e_{23}, x_{18}, x_{24}),
  h_{25}:\text{the}_q(x_{24}, h_{27}, h_{26}),
  h_{28}:\text{matter}\_\text{n}\_\text{of}(x_{24}, i_{29})
\end{array}
\{ h_{27} = q h_{28}, h_{21} = q h_{19}, h_{15} = q h_{16}, h_7 = q h_8, h_1 = q h_2 \}\rangle
\]
Approach

\{The German\} was sent for but professed to \{know\} \{nothing\} \{of the matter\}.

\langle h_{1}, e_{3},
  h_{4} : \text{the}_q(x_{6}, h_{7}, h_{5}),
  h_{8} : \text{named}(x_{6}, \text{German}),
  h_{9} : \text{send}_v\text{for}(e_{10}, i_{11}, x_{6}),
  h_{9} : \text{parg}_d(e_{12}, e_{10}, x_{6}),
  h_{2} : \text{but}_c(e_{3}, h_{9}, e_{10}, h_{14}, e_{13}),
  h_{14} : \text{profess}_v\text{to}(e_{13}, x_{6}, h_{15}),
  h_{16} : \text{know}_v\text{1}(e_{17}, x_{6}, x_{18}),
  h_{19} : \text{thing}(x_{18}),
  h_{20} : \text{no}_q(x_{18}, h_{21}, h_{22}),
  h_{19} : \text{of}_p(e_{23}, x_{18}, x_{24}),
  h_{25} : \text{the}_q(x_{24}, h_{27}, h_{26}),
  h_{28} : \text{matter}\_n\_of(x_{24}, i_{29})
\rangle

\{ h_{27} = q h_{28}, h_{21} = q h_{19}, h_{15} = q h_{16}, h_{7} = q h_{8}, h_{1} = q h_{2} \}
{The German} was sent for but professed to {know} ⟨nothing⟩ {of the matter}.

\[
\langle h_1, e_3, \\
  h_4: _{\text{the}_q}(0:3)(x_6, h_7, h_5), \\
  h_8: _{\text{named}}(4:10)(x_6, \text{German}), \\
  h_9: _{\text{send}_v_for}(15:19)(e_{10}, i_{11}, x_6), \\
  h_9: _{\text{parg}_d}(15:19)(e_{12}, e_{10}, x_6), \\
  h_2: _{\text{but}_c}(24:27)(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
  h_{14}: _{\text{profess}_v_to}(28:37)(e_{13}, x_6, h_{15}), \\
  h_{16}: _{\text{know}_v_1}(41:45)(e_{17}, x_6, x_{18}), \\
  h_{19}: _{\text{thing}}(46:53)(x_{18}), \\
  h_{20}: _{\text{no}_q}(46:53)(x_{18}, h_{21}, h_{22}), \\
  h_{19}: _{\text{of}_p}(54:56)(e_{23}, x_{18}, x_{24}), \\
  h_{25}: _{\text{the}_q}(57:60)(x_{24}, h_{27}, h_{26}), \\
  h_{28}: _{\text{matter}_n_of}(61:68)(x_{24}, i_{29})
\} = q h_{28}, h_{21} = q h_{19}, h_{15} = q h_{16}, h_7 = q h_8, h_1 = q h_2 \}
## Results

As of 2014, state of the art for this task

<table>
<thead>
<tr>
<th>Method</th>
<th>Scopes Prec</th>
<th>Scopes Rec</th>
<th>Scopes $F_1$</th>
<th>Tokens Prec</th>
<th>Tokens Rec</th>
<th>Tokens $F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read et al 2012</td>
<td>87.4</td>
<td>61.5</td>
<td>72.2</td>
<td>82.0</td>
<td>88.8</td>
<td>85.3</td>
</tr>
<tr>
<td>ERS Crawler</td>
<td>87.8</td>
<td>43.4</td>
<td>58.1</td>
<td>78.8</td>
<td>66.7</td>
<td>72.2</td>
</tr>
<tr>
<td>Combined System</td>
<td>87.6</td>
<td>62.7</td>
<td>73.1</td>
<td>82.6</td>
<td>88.5</td>
<td>85.4</td>
</tr>
</tbody>
</table>

Data/software for reproducibility: [http://www.delph-in.net/crawler/](http://www.delph-in.net/crawler/)
Task: Generate English from First-Order Logic

- Online course on introductory logic
- Students are presented with an English statement
- Their task: Produce an equivalent first-order logic expression
- Our task: Generate English paraphrases of an FOL
  - Produce English for auto-generated course FOL to start task
  - Restate student’s incorrect FOL as English for instruction
Our method

- Convert FOL to skeletal ERS (Python script)
- Inflate skeletal ERS to full ERS using ACE ‘transfer’ rules
- Apply richer set of transfer rules using ACE to produce paraphrase ERSs
- Generate from each of these paraphrase ERSs using ACE
- Select one of these outputs to present to the student
Example: FOL to English

First, convert FOL to skeletal ERS via Python script:

\[ \text{large}(a) \& \text{large}(b) \]

[ LTOP: h1
 INDEX: e1
 RELS: < [ "name" LBL: h3 ARG0: x1 CARG: "A" ]
 [ "large" LBL: h4 ARG0: e2 ARG1: x1 ]
 [ "name" LBL: h5 ARG0: x2 CARG: "B" ]
 [ "large" LBL: h6 ARG0: e3 ARG1: x2 ]
 [ "and" LBL: h2 ARG0: e1 L-INDEX: e2 R-INDEX: e3 ] > ]
‘Inflated’ ERS for large(a) & large(b)

Next, apply transfer rules to fill in missing elements (quantifiers, variable properties, ERS predicate names, handle constraints):

[ LTOP: h20
  RELS: < [ named LBL: h5 ARG0: x10 [ x PERS: 3 NUM: sg ] CARG: "A" ]
         [ named LBL: h9 ARG0: x11 [ x PERS: 3 NUM: sg ] CARG: "B" ]
         [ proper_q LBL: h2 ARG0: x10 RSTR: h3 BODY: h4 ]
         [ proper_q LBL: h6 ARG0: x11 RSTR: h7 BODY: h8 ]
         [ _and_c LBL: h12 ARG0: e13 L-INDEX: e14 R-INDEX: e15 L-HNDL: h16 R-HNDL: h17 ]
         [ _large_a_1 LBL: h18 ARG0: e14 [ e SF: prop TENSE: pres PERF: - ] ARG1: x10 ]
         [ _large_a_1 LBL: h19 ARG0: e15 [ e SF: prop TENSE: pres PERF: - ] ARG1: x11 ]>
  HCONS: < h3 qeq h5 h7 qeq h9 h16 qeq h18 h17 qeq h19 > ]
Paraphrase transfer rules

Then apply paraphrase transfer rules to produce multiple ERSs, and present each ERS to the generator.

Example rule for *B is large and C is large* $\rightarrow$ *B and C are large*

coord_subject_rule := openproof Omtr &
[ CONTEXT.RELS < [ PRED named, ARG0 x3 ],
  [ PRED named, ARG0 x6 ] >,
  INPUT.RELS < [ PRED _and_c, ARG0 e10, L-INDEX e2, R-INDEX e5 ],
  [ PRED pred1, ARG0 e2, ARG1 x3 ],
  [ PRED pred1, ARG0 e5, ARG1 x6 ] >
  OUTPUT.RELS < [ PRED _and_c, ARG0 x10, L-INDEX x3, R-INDEX x6 ],
  [ PRED pred1, ARG0 e10, ARG1 x10 ] >
Generated paraphrases

\[ \text{large}(a) \& \text{large}(b) \]

A is large and B is large.
A is large, and B is large.
A and B are large.
Both A and B are large.
A second example

(cube(a) & cube(b)) -> leftof(a, b)

If A is a cube and B is a cube, then A is to the left of B.
If A and B are cubes, then A is to the left of B.
If both A and B are cubes, then A is to the left of B.
If A and B are both cubes, then A is to the left of B.
A is to the left of B, if A and B are both cubes.
Task: Interpreting robotic spatial commands

- Semeval-2014 Shared Task 6
- Parse English commands to change states in a ‘blocks’ world
- Generate corresponding Robot Control Language statements
- Evaluate based on correct altered state of the game board
- [Packard, 2014]
Game board illustration
Example of robot command

Pick up the turquoise pyramid standing over a white cube

\[
\langle h_0, e_2, \\
h_4: \text{pronoun}_q(x_3, h_5, h_6), \\
h_7: \text{pron}(x_3), \\
h_1: \text{pick}_v\_up(e_2, x_3, x_8), \\
h_9: \text{the}_q(x_8, h_{10}, h_{11}), \\
h_{12}: \_\text{turquoise}_a\_1(e_{13}, x_8), \\
h_{12}: \_\text{pyramid}_n\_1(x_8), \\
h_{12}: \_\text{stand}_v\_1(e_{14}, x_8), \\
h_{12}: \_\text{over}_p(e_{15}, e_{14}, x_{16}), \\
h_{17}: \_\text{a}_q(x_{16}, h_{18}, h_{19}), \\
h_{20}: \_\text{white}_a\_1(e_{21}, x_{16}), \\
h_{20}: \_\text{cube}_n\_1(x_{16}) \\
\{ h_{18} = q h_{20}, h_{10} = q h_{12}, h_5 = q h_7, h_0 = q h_1 \} \rangle
\]
Generated robot command from ERS

*Pick up the turquoise pyramid standing over a white cube*

Corresponding RCL statement:

(event:
  (action: take)
  (entity:
    (id: 1)
    (color: cyan)
    (type: prism)
    (spatial-relation:
      (relation: above)
      (entity:
        (color: white)
        (type: cube)))
)
We are grateful to Emily Bender and Stephan Oepen for their considerable help in preparing these materials.
References I

Preprocessing and encoding techniques in PET.

The TreeBanker. A tool for supervised training of parsed corpora.

Implementing Typed Feature Structure Grammars.
CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, California.

Slacker semantics. Why superficiality, dependency and avoidance of commitment can be the right way to go.
In Proceedings of the 12th Meeting of the European Chapter of the Association for Computational Linguistics, page 1–9, Athens, Greece.


