English Resource Semantics

Dan Flickinger, Emily M. Bender & Woodley Packard

Stanford University & University of Washington

12 June 2016

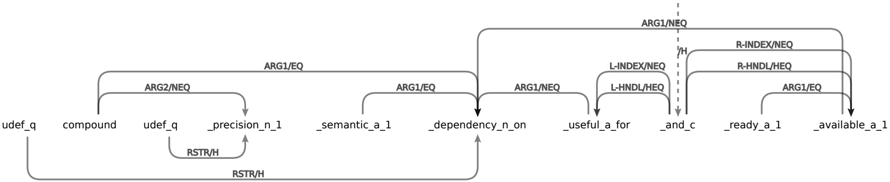
While you are waiting . . .

- English Resource Semantics: linguistically-motivated and useful!
- Software on USB drive, or downloadable: see `http://moin.delph-in.net/ErsTutorial`

Outline

What is an ERS?

- A rich, spanning, compositionally produced representation of sentence meaning: ‘who did what to whom’ (including control etc), grammatically constrained scope information, construction semantics.
- *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.
- Text(!) — ERS can be input to realization.
- Detailed documentation in progress at `http://moin.delph-in.net/ErgSemantics`

What can I do with an ERS?

Applications investigated include:

- Machine translation: e.g., [Bond et al., 2011]
- Information extraction and QA: e.g., [MacKinlay et al., 2009]
- Ontology extraction: e.g., [Nichols et al., 2005], [Herbelot and Copestake, 2006]
- Question generation: e.g., [Yao et al., 2012]
- Entailment recognition: e.g., [Lien and Kouylekov, 2014]
- Preprocessing for distributional semantics: e.g., [Herbelot, 2013]
- Detection scope of negation: e.g., [Packard et al., 2014]
- Robot control interface: e.g., [Packard, 2014a]
- Logic to English (for teaching logic)

What can I do with an ERS?

Applications investigated include:

- Machine translation: e.g., [Bond et al., 2011]
- Information extraction and QA: e.g., [MacKinlay et al., 2009]
- Ontology extraction: e.g., [Nichols et al., 2005], [Herbelot and Copestake, 2006]
- Question generation: e.g., [Yao et al., 2012]
- Entailment recognition: e.g., [Lien and Kouylekov, 2014]
- Preprocessing for distributional semantics: e.g., [Herbelot, 2013]
- Detection scope of negation: e.g., [Packard et al., 2014]
- Robot control interface: e.g., [Packard, 2014a]
- Logic to English (for teaching logic)

What can I do with an ERS?

Applications investigated include:

- Machine translation: e.g., [Bond et al., 2011]
- Information extraction and QA: e.g., [MacKinlay et al., 2009]
- Ontology extraction: e.g., [Nichols et al., 2005], [Herbelot and Copestake, 2006]
- Question generation: e.g., [Yao et al., 2012]
- Entailment recognition: e.g., [Lien and Kouylekov, 2014]
- Preprocessing for distributional semantics: e.g., [Herbelot, 2013]
- Detection scope of negation: e.g., [Packard et al., 2014]
- Robot control interface: e.g., [Packard, 2014a]
- Logic to English (for teaching logic)

How do I do things with an ERS?

Various open source tools:

- LOGON: `http://moin.delph-in.net/LogonTop`
  Experimentation with transfer-based MT (and other things)
- pyDelphin: `https://github.com/delph-in/pydelphin`
  An open source Python package implementing DELPH-IN representations, emphasis on MRS and DMRS support.
- pydmrs: a new Python toolkit for processing DMRS in various ways [Copestake et al., 2016]

Where does an ERS come from?

- The English Resource Grammar (ERG): 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 computational linguistics, 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 ACE `http://sweaglesw.org/linguistics/ace/`
  - With PET (included in the LOGON distribution) [Callmeier, 2002]
  - RESTful web service `http://moin.delph-in.net/ErgApi`
- Interactive single-sentence and batch parsing
- Software components with APIs for inclusion in NLP systems
- Online demonstrator `http://erg.delph-in.net/`

How can I get ERS?

- ERS-annotated sembanks (manually-verified analyses)

Treebank	Sents	Words	Domain
DeepBank	38766	756K	Wall Street Journal newspaper text (as in PTB)
LOGON	11041	149K	Tourism brochures
Verbmobil	11423	82K	Transcribed dialogues
WeScience	9835	152K	100 articles on NLP in Wikipedia
SemCor	2567	39K	Subset of Brown corpus, word-sense-annotated
SDP Brown	2243	33K	Balanced sample from Brown Corpus for SDP 2015
E-commerce	5429	46K	Customer emails
Misc	4277	66K	Test suites, essay, translations, online forum
Total	85581	1323K	

http://www.delph-in.net/redwoods

How can I get ERS?

- In a wide variety of formats:

  MRS Underspecified logical form with variables [Copestake, 2002, Copestake et al., 2005]

  DMRS Dependency MRS; Variable-free semantic dependency graph including scope [Copestake, 2009]

  EDS Elementary Dependency Structures; Variable-free semantic dependency graph without scope [Oepen and Lønning, 2006]

  DM Bilexical semantic dependencies; Only word-to-word dependencies [Ivanova et al., 2012]

- 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

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-1214.dat -1l`
- Type a simple test sentence, and hit Enter:
  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-1214.dat -1Tf` *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-1214.dat -1T 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.

$$
\begin{array}{l}
\langle\, h_1, e_3, \\
\left|\begin{array}{l}
h_4\text{:\_most\_q}(x_5, h_6, h_7), \\
h_8\text{:compound}(e_{10}, x_5, x_9), \\
h_{11}\text{:udef\_q}(x_9, h_{12}, h_{13}), \\
h_{14}\text{:\_house\_n\_of}(x_9, i_{15}), \\
h_8\text{:\_cat\_n\_1}(x_5), \\
h_2\text{:\_easy\_a\_for}(e_3, h_{16}, x_{17}), \\
h_{18}\text{:udef\_q}(x_{17}, h_{19}, h_{20}), \\
h_{21}\text{:\_dog\_n\_1}(x_{17}), \\
h_{22}\text{:\_chase\_v\_1}(e_{23}, x_{17}, x_5)
\end{array}\right| \\
\{\, h_1 =_q h_2,\, h_6 =_q h_8,\, h_{12} =_q h_{14},\, h_{16} =_q h_{22},\, h_{19} =_q h_{21} \,\}\,\rangle
\end{array}
$$

Example sentence 1

Most house cats are easy for dogs to chase.

$$\langle\, h_1, e_3,$$

$$\left|\begin{array}{l} h_4\text{:\_most\_q}(x_5, h_6, h_7), \\ h_8\text{:compound}(e_{10}, x_5, x_9), \\ h_{11}\text{:udef\_q}(x_9, h_{12}, h_{13}), \\ h_{14}\text{:\_house\_n\_of}(x_9, i_{15}), \\ h_8\text{:\_cat\_n\_1}(x_5), \\ h_2\text{:\_easy\_a\_for}(e_3, h_{16}, x_{17}), \\ h_{18}\text{:udef\_q}(x_{17}, h_{19}, h_{20}), \\ h_{21}\text{:\_dog\_n\_1}(x_{17}), \\ h_{22}\text{:\_chase\_v\_1}(e_{23}, x_{17}, x_5) \end{array}\right|$$

$$\{\, h_1 =_q h_2,\ h_6 =_q h_8,\ h_{12} =_q h_{14},\ h_{16} =_q h_{22},\ h_{19} =_q h_{21} \,\}\,\rangle$$

Example sentence 2

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

$$\langle\, h_1, e_3,$$

$$\left|\begin{array}{l}
h_4\text{:_which_q}(x_5, h_6, h_7),\\
h_8\text{:_book_n_of}(x_5, i_9),\\
h_{10}\text{:_the_q}(x_{12}, h_{13}, h_{11}),\\
h_{14}\text{:_guy_n_1}(x_{12}),\\
h_{14}\text{:_leave_v_1}(e_{15}, x_{12}, i_{16}),\\
h_2\text{:_give_v_1}(e_3, x_{12}, x_5, x_{17}),\\
h_{18}\text{:def_explicit_q}(x_{17}, h_{20}, h_{19}),\\
h_{21}\text{:poss}(e_{23}, x_{17}, x_{22}),\\
h_{24}\text{:pronoun_q}(x_{22}, h_{25}, h_{26}),\\
h_{27}\text{:pron}(x_{22}),\\
h_{21}\text{:_neighbor_n_1}(x_{17})
\end{array}\right|$$

$$\{\, h_1 =_q h_2,\ h_6 =_q h_8,\ h_{13} =_q h_{14},\ h_{20} =_q h_{21},\ h_{25} =_q h_{27} \,\}\,\rangle$$

Example sentence 2

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

$$\langle h_1, e_3,$$

h_4:_which_q(x_5, h_6, h_7),
h_8:_book_n_of(x_5, i_9),
h_{10}:_the_q(x_{12}, h_{13}, h_{11}),
h_{14}:_guy_n_1(x_{12}),
h_{14}:_leave_v_1(e_{15}, x_{12}, i_{16}),
h_2:_give_v_1(e_3, x_{12}, x_5, x_{17}),
h_{18}:def_explicit_q(x_{17}, h_{20}, h_{19}),
h_{21}:poss(e_{23}, x_{17}, x_{22}),
h_{24}:pronoun_q(x_{22}, h_{25}, h_{26}),
h_{27}:pron(x_{22}),
h_{21}:_neighbor_n_1(x_{17})

$$\{ h_1 =_q h_2, h_6 =_q h_8, h_{13} =_q h_{14}, h_{20} =_q h_{21}, h_{25} =_q h_{27} \} \rangle$$

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.

$$\langle\, h_1,\, e_3,\ \left|\begin{array}{l} h_4\text{:\_the\_q}(x_6,\, h_7,\, h_5), \\ h_8\text{:\_cat\_n\_1}(x_6), \\ h_2\text{:\_sleep\_v\_1}(e_3,\, x_6) \end{array}\right|\ \{\, h_1 =_q h_2,\ h_7 =_q h_8 \,\}\, \rangle$$

Introduction to ERS formalism

The cat sleeps.

$$\langle\, h_1, e_3, \left| \begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:\_cat\_n\_1}(x_6), \\ h_2\text{:\_sleep\_v\_1}(e_3, x_6) \end{array} \right| \{\, h_1 =_q h_2,\ h_7 =_q h_8 \,\} \,\rangle$$

- Top handle

Introduction to ERS formalism

The cat sleeps.

$$\langle\, h_1, e_3, \left|\begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:\_cat\_n\_1}(x_6), \\ h_2\text{:\_sleep\_v\_1}(e_3, x_6) \end{array}\right| \{\, h_1 =_q h_2, h_7 =_q h_8 \,\} \,\rangle$$

- Top handle
- Index

Introduction to ERS formalism

The cat sleeps.

$$\langle\, h_1, e_3,\\ \left|\begin{array}{l} h_4{:}\_\text{the\_q}(x_6, h_7, h_5), \\ h_8{:}\_\text{cat\_n\_1}(x_6), \\ h_2{:}\_\text{sleep\_v\_1}(e_3, x_6) \end{array}\right|\\ \{\, h_1 =_q h_2,\; h_7 =_q h_8 \,\}\,\rangle$$

- Top handle
- Index
- Bag of elementary predications

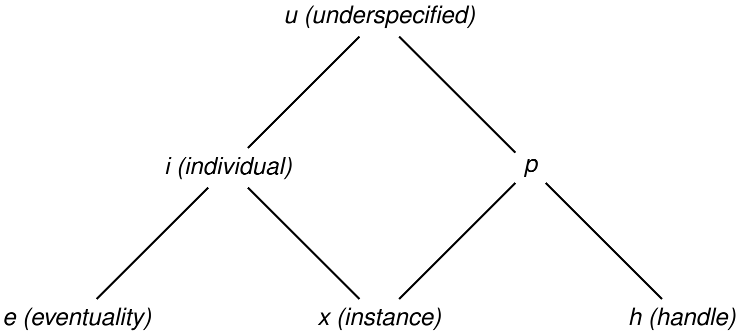
Introduction to ERS formalism

The cat sleeps.

$$\langle\, h_1, e_3, \left| \begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:\_cat\_n\_1}(x_6), \\ h_2\text{:\_sleep\_v\_1}(e_3, x_6) \end{array} \right| \{\, h_1 =_q h_2,\ h_7 =_q h_8 \,\} \,\rangle$$

- Top handle
- Index
- Bag of elementary predications
- Scope constraints

ERS variable types



Properties of variables

- Number, person, gender, and individuation on instances

  h_8:_cat_n_1(ARG0 x_6\{PERS *3*, NUM *sg*, GEND *n*, IND *+*\})

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

  h_2:_sleep_v_1(
  ARG0 e_3\{SF *prop*, TENSE *pres*, MOOD *indicative*, PROG -, PERF -\},
  ARG1 x_6)

Elementary predications

Every predication contains

- Predicate name
- Label of type *handle*
- Intrinsic argument of type *individual* as ARG0
  (except that the ARG0 of quantifiers is not intrinsic)

Predications may contain additional arguments, as (mostly) ARG1, ARG2, ..., though quantifiers and conjunctions, among others, use a richer inventory of argument names.

Scope constraints

The cat sleeps.

$$\langle\, h_1,\, e_3,\ \left|\begin{array}{l} h_4{:}\_the\_q(x_6,\, h_7,\, h_5), \\ h_8{:}\_cat\_n\_1(x_6), \\ h_2{:}\_sleep\_v\_1(e_3,\, x_6) \end{array}\right|\ \{\, h_1 =_q h_2,\ h_7 =_q h_8 \,\}\,\rangle$$

- Equivalent to: _the_q(x_6, _cat_n_1(x_6), _sleep_v_1(e_3, x_6))
- The scope constraints indicate how the EPs fit together to give the fully scoped logical form.
- MRS is underspecified: usually *many* logical forms (roughly n!, where n is the number of NPs in the sentence).

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
      (number or string, e.g.: tile_n_1, break_v_cause)
- **Abstract** predicates are introduced either via construction, or in decomposed semantics of lexical entries.
  - Examples: compound, ellipsis, superl

Abstract predicate example: Noun-noun compounds

The **police dog** *barked.*

$$\langle\, h_1, e_3, \\ \left|\begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:compound}(e_{10}, x_6, x_9), \\ h_{11}\text{:udef\_q}(x_9, h_{12}, h_{13}), \\ h_{14}\text{:\_police\_n\_1}(x_9), \\ h_8\text{:\_dog\_n\_1}(x_6), \\ h_2\text{:\_bark\_v\_1}(e_3, x_6) \end{array}\right| \\ \{\, h_1 =_q h_2,\, h_7 =_q h_8,\, h_{12} =_q h_{14} \,\}\,\rangle$$

Parameterized predications

Words for named entities introduce in their semantic predication a parameter as the value of a distinguished attribute CARG

We admire Kim greatly.

$$h_{13}\text{:named}(x_9, \textit{Kim})$$

Scopal arguments

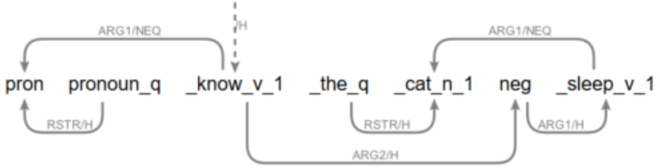
A predication may have a handle as the value of one of its argument attributes, with a corresponding handle constraint identifying the label of the highest-scoping predication of the argument phrase.

We **know** *that the cat* **didn't sleep**.

$$\langle\, h_1, e_3, \left| \begin{array}{l} h_4\text{:pron}(x_5), \\ h_6\text{:pronoun\_q}(x_5, h_7, h_8), \\ h_2\text{:\_know\_v\_1}(e_3, x_5, h_9), \\ h_{10}\text{:\_the\_q}(x_{12}, h_{13}, h_{11}), \\ h_{14}\text{:\_cat\_n\_1}(x_{12}), \\ h_{15}\text{:neg}(e_{17}, h_{16}), \\ h_{18}\text{:\_sleep\_v\_1}(e_{19}, x_{12}) \end{array} \right| \{\, h_1 =_q h_2,\ h_7 =_q h_4,\ h_9 =_q h_{15},\ h_{13} =_q h_{14},\ h_{16} =_q h_{18} \,\} \rangle$$

Scopal arguments in other formats

DMRS:



Scoped form:

pronoun_q(x,pron(x),the(y,cat(y),know(e,x,neg(sleep(e1,y)))))

Plus other scoped structures, but these are all logically equivalent in this example.

Basic assumptions for well-formed ERS

- Every predication that isn't a quantifier has a unique 'intrinsic' ARG0
- 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

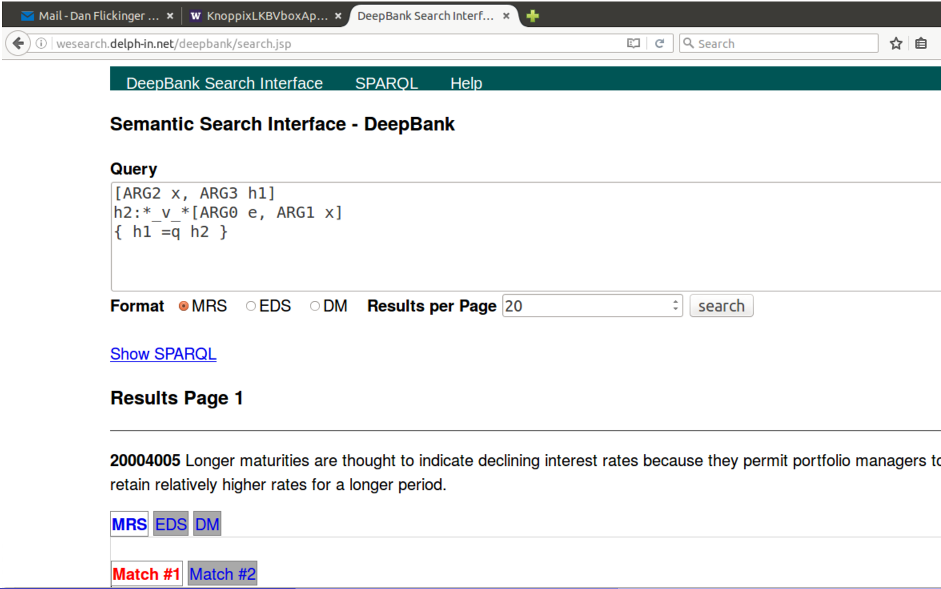
Introduction to the treebanks

- Several collections of text in a variety of domains
- 85,000 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
- Software support for conversion and export to multitude of formats

Semantic search via fingerprints

- Identify elements of ERS to match in treebank
- Query by example: partial, ‘annotated’ sub-structures
- Returns sentences and their ERS (in multiple views)
- Useful for exploring ERS in support of feature design

Fingerprint search example: ‘Object’ Control



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

- Documentation of query language:
  `http://moin.delph-in.net/WeSearch/QueryLanguage`
- 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
- DM bi-lexical dependencies
- Direct ERS output from ACE

Standard MRS (terse)

The jungle lion was chasing a small giraffe.

$$
\langle\, h_1, e_3,\\
\left|\begin{array}{l}
h_4\text{:\_the\_q}(x_6, h_7, h_5),\\
h_8\text{:compound}(e_{10}, x_6, x_9),\\
h_{11}\text{:udef\_q}(x_9, h_{12}, h_{13}),\\
h_{14}\text{:\_jungle\_n\_1}(x_9),\\
h_8\text{:\_lion\_n\_1}(x_6),\\
h_2\text{:\_chase\_v\_1}(e_3, x_6, x_{15}),\\
h_{16}\text{:\_a\_q}(x_{15}, h_{18}, h_{17}),\\
h_{19}\text{:\_small\_a\_1}(e_{20}, x_{15}),\\
h_{19}\text{:\_giraffe\_n\_1}(x_{15})
\end{array}\right|\\
\{\, h_1 =_q h_2, h_7 =_q h_8, h_{12} =_q h_{14}, h_{18} =_q h_{19} \,\}\,\rangle
$$

Standard MRS with argument roles

The jungle lion was chasing a small giraffe.

$$\langle\, h_1, e_3,$$

$$\left|\begin{array}{l} h_4\text{:\_the\_q(ARG0 } x_6\text{, RSTR } h_7\text{, BODY } h_5\text{),} \\ h_8\text{:compound(ARG0 } e_{10}\text{, ARG1 } x_6\text{, ARG2 } x_9\text{),} \\ h_{11}\text{:udef\_q(ARG0 } x_9\text{, RSTR } h_{12}\text{, BODY } h_{13}\text{),} \\ h_{14}\text{:\_jungle\_n\_1(ARG0 } x_9\text{),} \\ h_8\text{:\_lion\_n\_1(ARG0 } x_6\text{),} \\ h_2\text{:\_chase\_v\_1(}\textbf{ARG0 } e_3\text{, }\textbf{ARG1 } x_6\text{, }\textbf{ARG2 } x_{15}\text{),} \\ h_{16}\text{:\_a\_q(ARG0 } x_{15}\text{, RSTR } h_{18}\text{, BODY } h_{17}\text{),} \\ h_{19}\text{:\_small\_a\_1(ARG0 } e_{20}\text{, ARG1 } x_{15}\text{),} \\ h_{19}\text{:\_giraffe\_n\_1(ARG0 } x_{15}\text{)} \end{array}\right|$$

$$\{\, h_1 =_q h_2,\ h_7 =_q h_8,\ h_{12} =_q h_{14},\ h_{18} =_q h_{19} \,\}\,\rangle$$

Standard MRS with argument roles and properties

The jungle lion was chasing a small giraffe.

$$
\begin{array}{l}
\langle\, h_1, e_3, \\
\quad h_4\text{:\_the\_q(ARG0 } x_6, \quad , \\
\qquad\qquad \text{RSTR } h_7, \text{ BODY } h_5) \\
\quad h_8\text{:compound(ARG0 } e_{10}\{\text{SF } \textit{prop}, \text{TENSE } \textit{untensed}, \text{MOOD } \textit{indic}, \text{PROG -}, \text{PERF -}\}, \\
\qquad\qquad \text{ARG1 } x_6, \text{ ARG2 } x_9\{\text{IND +}\}) \\
\quad h_{11}\text{:udef\_q(ARG0 } x_9, \text{ RSTR } h_{12}, \text{ BODY } h_{13}), \\
\quad h_{14}\text{:\_jungle\_n\_1(ARG0 } x_9), \\
\quad h_8\text{:\_lion\_n\_1(ARG0 } x_6\{\textbf{PERS } \textit{3}, \textbf{NUM } \textit{sg}, \textbf{IND } +\}), \\
\quad h_2\text{:\_chase\_v\_1(ARG0 } e_3\{\textbf{SF } \textit{prop}, \textbf{TENSE } \textit{past}, \textbf{MOOD } \textit{indic}, \textbf{PROG } +, \textbf{PERF } -\}, \\
\qquad\qquad \text{ARG1 } x_6, \text{ ARG2 } x_{15}\{\text{PERS } \textit{3}, \text{NUM } \textit{sg}, \text{IND } +\}) \\
\quad h_{16}\text{:\_a\_q(ARG0 } x_{15}, \text{ RSTR } h_{18}, \text{ BODY } h_{17}), \\
\quad h_{19}\text{:\_small\_a\_1(ARG0 } e_{20}\{\text{SF } \textit{prop}, \text{TENSE } \textit{untensed}, \text{MOOD } \textit{indic}\}, \text{ ARG1 } x_{15}), \\
\quad h_{19}\text{:\_giraffe\_n\_1(ARG0 } x_{15}) \\
\{\, h_1 =_q h_2,\ h_7 =_q h_8,\ h_{12} =_q h_{14},\ h_{18} =_q h_{19} \,\}\,\rangle
\end{array}
$$

Standard MRS also with character positions

The jungle lion was chasing a small giraffe.

$$
\langle\, h_1, e_3,\\
\left|\begin{array}{l}
h_4\text{:\_the\_q}\langle 0\,{:}\,3\rangle(\text{ARG0 } x_6,\ \text{RSTR } h_7,\ \text{BODY } h_5),\\
h_8\text{:compound}\langle \mathbf{4}\,{:}\,\mathbf{15}\rangle(\text{ARG0 } e_{10},\ \text{ARG1 } x_6,\ \text{ARG2 } x_9),\\
h_{11}\text{:udef\_q}\langle 4\,{:}\,10\rangle(\text{ARG0 } x_9,\ \text{RSTR } h_{12},\ \text{BODY } h_{13}),\\
h_{14}\text{:\_jungle\_n\_1}\langle \mathbf{4}\,{:}\,\mathbf{10}\rangle(\text{ARG0 } x_9),\\
h_8\text{:\_lion\_n\_1}\langle \mathbf{11}\,{:}\,\mathbf{15}\rangle(\text{ARG0 } x_6),\\
h_2\text{:\_chase\_v\_1}\langle 20\,{:}\,27\rangle(\text{ARG0 } e_3,\ \text{ARG1 } x_6,\ \text{ARG2 } x_{15}),\\
h_{16}\text{:\_a\_q}\langle 28\,{:}\,29\rangle(\text{ARG0 } x_{15},\ \text{RSTR } h_{18},\ \text{BODY } h_{17}),\\
h_{19}\text{:\_small\_a\_1}\langle 30\,{:}\,35\rangle(\text{ARG0 } e_{20},\ \text{ARG1 } x_{15}),\\
h_{19}\text{:\_giraffe\_n\_1}\langle 36\,{:}\,44\rangle(\text{ARG0 } x_{15})
\end{array}\right|\\
\{\, h_1 =_q h_2,\ h_7 =_q h_8,\ h_{12} =_q h_{14},\ h_{18} =_q h_{19} \,\}\,\rangle
$$

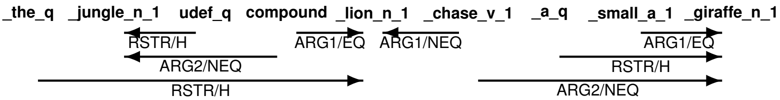
Simple MRS (textual exchange format)

The jungle lion was chasing a small giraffe.

```
[ LTOP   [h1] h
  INDEX  [e3] e
  RELS  < [ LBL [h4] h    PRED _the_q    ARG0 [x6] x   RSTR [h7] h   BODY [h5] h ],
          [ LBL [h8] h    PRED compound  ARG0 [e10] e  ARG1 [x6]     ARG2 [x9] x ],
          [ LBL [h11] h   PRED udef_q    ARG0 [x9]     RSTR [h12] h  BODY [h13] h ],
          [ LBL [h14] h   PRED _jungle_n_1  ARG0 [x9] ],
          [ LBL [h8]      PRED _lion_n_1    ARG0 [x6] ],
          [ LBL [h2] h    PRED _chase_v_1   ARG0 [e3]  ARG1 [x6]  ARG2 [x15] x ],
          [ LBL [h16] h   PRED _a_q      ARG0 [x15]    RSTR [h18] h  BODY [h17] h ],
          [ LBL [h19] h   PRED _small_a_1   ARG0 [e20] e  ARG1 [x15] ],
          [ LBL [h19]     PRED _giraffe_n_1 ARG0 [x15] ] >
  HCONS < [ HARG [h1]  LARG [h2] ], [ HARG [h7]  LARG [h8] ],
          [ HARG [h12] LARG [h14] ], [ HARG [h18] LARG [h19] ] > ]
```

DMRS

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.

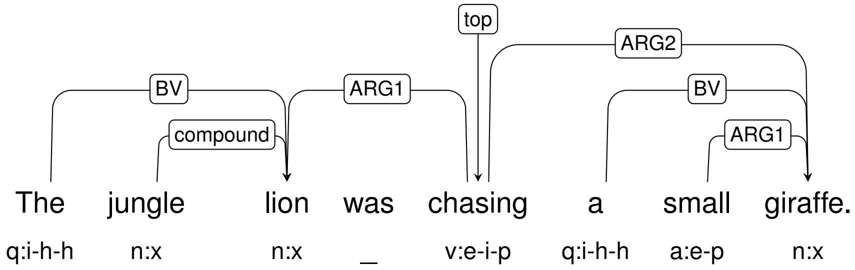
The jungle lion was chasing a small giraffe.

```
(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. (2016): *Comparability of Linguistic Graph Banks*.



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]
[compound_rel<4:15> LBL: h7 ARG0: e8 [e SF: prop TENSE: untensed MOOD: indicative PROG: - PERF: -] ARG1: x3 ARG2: x9 [x IND: +]]
[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 >]

DMRS XML output

The jungle lion was chasing a small giraffe.


```
<dmrs>
<node nodeid='10001' cfrom='0' cto='3'><gpred>_the_q</gpred><sortinfo cvarsort='x' pers='3' num='sg' ind='plus'/></node>
<node nodeid='10002' cfrom='4' cto='15'><gpred>compound</gpred><sortinfo cvarsort='e' sf='prop' tense='untensed'
 mood='indicative' prog='minus' perf='minus'/></node>
<node nodeid='10003' cfrom='4' cto='10'><gpred>udef_q</gpred><sortinfo cvarsort='x' ind='plus'/></node>
<node nodeid='10004' cfrom='4' cto='10'><gpred>_jungle_n_1</gpred><sortinfo cvarsort='x' ind='plus'/></node>
<node nodeid='10005' cfrom='11' cto='15'><gpred>_lion_n_1</gpred><sortinfo cvarsort='x' pers='3' num='sg' ind='plus'/></node>
<node nodeid='10006' cfrom='20' cto='27'><gpred>_chase_v_1</gpred><sortinfo cvarsort='e' sf='prop' tense='past'
 mood='indicative' prog='plus' perf='minus'/></node>
<node nodeid='10007' cfrom='28' cto='29'><gpred>_a_q</gpred><sortinfo cvarsort='x' pers='3' num='sg' ind='plus'/></node>
<node nodeid='10008' cfrom='30' cto='35'><gpred>_small_a_1</gpred><sortinfo cvarsort='e' sf='prop' tense='untensed'
 mood='indicative'/></node>
<node nodeid='10009' cfrom='36' cto='44'><gpred>_giraffe_n_1</gpred><sortinfo cvarsort='x' pers='3' num='sg' ind='plus'/></node>
<link from='10001' to='10002'><rargname>RSTR</rargname><post>H</post></link>
<link from='10001' to='10005'><rargname>RSTR</rargname><post>H</post></link>
<link from='10002' to='10001'><rargname>ARG1</rargname><post>NEQ</post></link>
<link from='10002' to='10003'><rargname>ARG2</rargname><post>NEQ</post></link>
<link from='10002' to='10005'><rargname>NIL</rargname><post>EQ</post></link>
<link from='10003' to='10004'><rargname>RSTR</rargname><post>H</post></link>
<link from='10006' to='10001'><rargname>ARG1</rargname><post>NEQ</post></link>
<link from='10006' to='10007'><rargname>ARG2</rargname><post>NEQ</post></link>
<link from='10007' to='10008'><rargname>RSTR</rargname><post>H</post></link>
<link from='10007' to='10009'><rargname>RSTR</rargname><post>H</post></link>
<link from='10008' to='10007'><rargname>ARG1</rargname><post>NEQ</post></link>
<link from='10008' to='10009'><rargname>NIL</rargname><post>EQ</post></link>
</dmrs>
```

Inspection and conversion tools

- LUI: inspection
- pyDelphin: conversion and inspection
  `https://github.com/delph-in/pydelphin`

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

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.

$$
\langle\, h_1, e_3, \left|\begin{array}{l}
h_2\text{:\_seem\_v\_to}(e_3, h_4, i_5), \\
h_6\text{:proper\_q}(x_8, h_7, h_9), \\
h_{10}\text{:named}(x_8, \textit{Kim}), \\
h_{11}\text{:\_go\_v\_1}(e_{12}, x_8), \\
h_{13}\text{:\_and\_c}(e_{14}, h_{11}, e_{12}, h_{15}, e_{16}), \\
h_{15}\text{:\_rely\_v\_on}(e_{16}, x_8, x_{17}), \\
h_{18}\text{:proper\_q}(x_{17}, h_{19}, h_{20}), \\
h_{21}\text{:named}(x_{17}, \textit{Sandy})
\end{array}\right| \{ h_{19} =_q h_{21}, h_7 =_q h_{10}, h_4 =_q h_{13}, h_1 =_q h_2 \} \,\rangle
$$

Not all surface words are directly reflected in the ERS

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

$$\langle\, h_1, e_3,$$
$$h_2\text{:\_seem\_v\_to}(e_3, h_4, i_5),$$
$$h_6\text{:proper\_q}(x_8, h_7, h_9),$$
$$h_{10}\text{:named}(x_8, \textit{Kim}),$$
$$h_{11}\text{:\_go\_v\_1}(e_{12}, x_8),$$
$$h_{13}\text{:\_and\_c}(e_{14}, h_{11}, e_{12}, h_{15}, e_{16}),$$
$$h_{15}\text{:\_rely\_v\_on}(e_{16}, x_8, x_{17}),$$
$$h_{18}\text{:proper\_q}(x_{17}, h_{19}, h_{20}),$$
$$h_{21}\text{:named}(x_{17}, \textit{Sandy})$$
$$\{\, h_{19} =_q h_{21}, h_7 =_q h_{10}, h_4 =_q h_{13}, h_1 =_q h_2 \,\}\,\rangle$$

Sentential negation analyzed in terms of the scopal predicate **neg**

The dog didn't bark.

$$\langle\, h_1, e_3,\ \left|\ \begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:\_dog\_n\_1}(x_6), \\ h_2\text{:neg}(e_{10}, h_9), \\ h_{11}\text{:\_bark\_v\_1}(e_3, x_6) \end{array}\ \right|\ \{\, h_9 =_q h_{11},\ h_7 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle$$

Contracted negation (*didn't, won't*) and independent *not* normalized

The dog did not bark.

$$\langle\, h_1, e_3, \left|\begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:\_dog\_n\_1}(x_6), \\ h_2\text{:neg}(e_{10}, h_9), \\ h_{11}\text{:\_bark\_v\_1}(e_3, x_6) \end{array}\right| \{\, h_9 =_q h_{11}, h_7 =_q h_8, h_1 =_q h_2 \,\}\,\rangle$$

Scope of negation fixed by grammatical constraints

Sandy knows that Kim probably didn't leave.

$$\langle\, h_1, e_3,$$

$$\left|\begin{array}{l} h_4\text{:proper\_q}(x_6, h_5, h_7), \\ h_8\text{:named}(x_6, \textit{Sandy}), \\ h_2\text{:\_know\_v\_1}(e_3, x_6, h_9), \\ h_{10}\text{:proper\_q}(x_{12}, h_{11}, h_{13}), \\ h_{14}\text{:named}(x_{12}, \textit{Kim}), \\ h_{15}\text{:\_probable\_a\_1}(e_{16}, h_{17}), \\ h_{18}\text{:neg}(e_{20}, h_{19}), \\ h_{21}\text{:\_leave\_v\_1}(e_{22}, x_{12}, p_{23}) \end{array}\right|$$

$$\{\, h_{19} =_q h_{21},\ h_{17} =_q h_{18},\ h_{11} =_q h_{14},\ h_9 =_q h_{15},\ h_5 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle$$

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.

$$\langle h_1, e_3,$$

$$\left| \begin{array}{l} h_4\text{:proper\_q}(x_6, h_5, h_7), \\ h_8\text{:named}(x_6, Kim), \\ h_2\text{:\_probable\_a\_1}(e_9, h_{10}), \\ h_{11}\text{:\_see\_v\_1}(e_3, x_6, x_{12}), \\ h_{13}\text{:\_no\_q}(x_{12}, h_{15}, h_{14}), \\ h_{16}\text{:\_dog\_n\_1}(x_{12}) \end{array} \right|$$

$$\{ h_{15} =_q h_{16}, h_{10} =_q h_{11}, h_5 =_q h_8, h_1 =_q h_2 \} \rangle$$

Morphological negation unanalyzed (for now)

That dog is invisible.

$$\langle\, h_1, e_3,$$
$$\left|\begin{array}{l} h_4\text{:\_that\_q\_dem}(x_6, h_7, h_5), \\ h_8\text{:\_dog\_n\_1}(x_6), \\ h_2\text{:\_invisible\_a\_to}(e_3, x_6, i_9) \end{array}\right|$$
$$\{\, h_7 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle$$

Lexically negative verbs not decomposed

The dog failed to bark.

$$\langle\, h_1, e_3,\ \left|\begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:\_dog\_n\_1}(x_6), \\ h_2\text{:\_fail\_v\_1}(e_3, h_9), \\ h_{10}\text{:\_bark\_v\_1}(e_{11}, x_6) \end{array}\right| \ \{\, h_9 =_q h_{10},\ h_7 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle$$

Negation interacts with the analysis of sentence fragments

Not this year.

$$\langle\, h_1, e_3,$$
$$\left|\begin{array}{l} h_2\text{:unknown}(e_3, u_4), \\ h_2\text{:neg}(e_6, h_5), \\ h_7\text{:loc\_nonsp}(e_8, e_3, x_9), \\ h_{10}\text{:\_this\_q\_dem}(x_9, h_{12}, h_{11}), \\ h_{13}\text{:\_year\_n\_1}(x_9) \end{array}\right|$$
$$\{\, h_{12} =_q h_{13},\ h_5 =_q h_7,\ h_1 =_q h_2 \,\}\,\rangle$$

Negation fingerprints

```
neg[ARG1 h1]
h2:[ARG0 e]
{ h1 =q h2 }
```

Some predicates establish required coreference relations

Kim persuaded Sandy to leave.

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

Which arguments are shared is predicate-specific

Kim promised Sandy to leave.

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

Control predicates: Not just verbs

Kim is happy to leave.

$$\langle\, h_1, e_3, \left| \begin{array}{l} h_4\text{:proper\_q}(x_6, h_5, h_7), \\ h_8\text{:named}(x_6, \mathit{Kim}), \\ h_2\text{:\_happy\_a\_with}(e_3, x_6, h_9), \\ h_{10}\text{:\_leave\_v\_1}(e_{11}, x_6, p_{12}) \end{array} \right| \{\, h_9 =_q h_{10},\, h_5 =_q h_8,\, h_1 =_q h_2 \,\} \,\rangle$$

Control predicates involve diverse syntactic frames; normalized at the semantic level

Kim prevented Sandy from leaving.

$$\langle\, h_1, e_3,$$

$$\left|\begin{array}{l} h_4\text{:proper\_q}(x_6, h_5, h_7), \\ h_8\text{:named}(x_6, \textit{Kim}), \\ h_2\text{:\_prevent\_v\_from}(e_3, x_6, x_{10}, h_9), \\ h_{11}\text{:proper\_q}(x_{10}, h_{12}, h_{13}), \\ h_{14}\text{:named}(x_{10}, \textit{Sandy}), \\ h_{15}\text{:\_leave\_v\_1}(e_{16}, x_{10}, p_{17}) \end{array}\right|$$

$$\{\, h_{12} =_q h_{14}, h_9 =_q h_{15}, h_5 =_q h_8, h_1 =_q h_2 \,\}\,\rangle$$

Control fingerprints

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

```
[ARG0 e1, ARG2 x2, ARG3 h3]
h4:[ARG0 e5, ARG1 x2]
{ h3 =q h4 }
```

Lexically Mediated

Complex examples are easy to find.

$$\langle\, h_1, e_3,$$
$$h_4\text{:udef\_q}(x_6, h_5, h_7),$$
$$h_8\text{:\_complex\_a\_1}(e_9, x_6),$$
$$h_8\text{:\_example\_n\_of}(\color{red}{x_6}, i_{10}),$$
$$h_2\text{:\_easy\_a\_for}(e_3, h_{11}, i_{12}),$$
$$h_{13}\text{:\_find\_v\_1}(e_{14}, i_{12}, \color{red}{x_6})$$
$$\{\, h_{11} =_q h_{13},\ h_5 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle$$

Relative clauses

The cat whose collar you thought I found escaped.

$$\langle\, h_1, e_3,$$
$$\left|\begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:\_cat\_n\_1}(x_6), \\ h_9\text{:def\_explicit\_q}(x_{11}, h_{12}, h_{10}), \\ h_{13}\text{:poss}(e_{14}, x_{11}, x_6), \\ h_{15}\text{:\_collar\_n\_1}(x_{11}), \\ h_{16}\text{:pron}(x_{17}), \\ h_{18}\text{:pronoun\_q}(x_{17}, h_{19}, h_{20}), \\ h_8\text{:\_think\_v\_1}(e_{21}, x_{17}, h_{23}, i_{22}), \\ h_{24}\text{:pron}(x_{25}), \\ h_{26}\text{:pronoun\_q}(x_{25}, h_{27}, h_{28}), \\ h_{29}\text{:\_find\_v\_1}(e_{30}, x_{25}, x_{11}), \\ h_2\text{:\_escape\_v\_1}(e_3, x_6, p_{31}) \end{array}\right|$$
$$\{\, h_{27} =_q h_{24},\ h_{23} =_q h_{29},\ h_{19} =_q h_{16},\ h_{12} =_q h_{15},\ h_7 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle$$

Right Node Raising

PCBs move into and go out of the machine automatically.

$$\langle\, h_1, e_{10},$$
$$\begin{array}{|l|}
h_4\text{:udef_q}(x_6, h_5, h_7),\\
h_8\text{:_pcbs/nns_u_unknown}(x_6),\\
h_9\text{:_move_v_1}(e_{10}, x_6),\\
h_9\text{:_into_p}(e_{11}, e_{10}, \color{red}{x_{12}}),\\
h_2\text{:_and_c}(e_3, h_9, e_{10}, h_{14}, e_{13}),\\
h_{14}\text{:_go_v_1}(e_{13}, x_6),\\
h_{14}\text{:_out+of_p_dir}(e_{15}, e_{13}, \color{red}{x_{12}}),\\
h_{16}\text{:_the_q}(x_{12}, h_{18}, h_{17}),\\
h_{19}\text{:_machine_n_1}(\color{red}{x_{12}}),\\
h_2\text{:_automatic_a_1}(e_{20}, e_3)
\end{array}$$
$$\{\, h_{18} =_q h_{19},\ h_5 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle$$

Fingerprints?

- 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

Parser settings

- ACE invocation flags
- Root symbols
- Preprocessing
- Unknown-word handling
- Disambiguation models
- Resource limits

Parser settings: ACE invocation flags

- `-g erg/erg-1214.dat` – what grammar to use
- `-1` – how many results to show (or `-n 10`)
- `-T` – suppress printing the derivation tree
- `-f` – pretty-print the ERS with one predication per line
- `--show-probability` – print the probability of each parse
- rebuild grammar file (after changing config.tdl):
  `ace -G my-erg-1214.dat -g erg/ace/config.tdl`

Parser settings: ACE generator

- Option `-e` – generate from an input ERS
- Try, for example, the following command to have the parser produce an ERS and then generate from it:
- `echo “If cats run, dogs won’t bark.” | ace -g erg/erg-1214.dat -1 | ace -g erg/erg-1214.dat -e`

Parser settings: root symbols

- `ace -g erg/erg-1214.dat -1Tf -r "root1 root2 root3..."`
- erg/ace/config.tdl: `parsing-roots := root1 root2 root3.`
- root_strict *Kim stole the cookie.*
- root_informal *Kim stole, the cookie*
- root_frag *The cookie that Kim stole.*
- root_inffrag *The cookie that Kim, stole.*
- root_robust *Kim stole the the cookie.*

Parser settings: preprocessing

- REPP modes:
  - erg/ace/config.tdl: `preprocessor-modules := mod1 mod2 mod3.`
  - ../rpp/xml.rpp, ../rpp/ascii.rp, ../rpp/quotes.rpp
    – Unicode-ify various ASCII conventions
  - ../rpp/html.rpp
    – strip simple (by no means all) HTML markup from input
  - ../rpp/wiki.rpp – strip Wikipedia markup from input
  - ../rpp/gml.rpp – “Grammatical Markup Language” for selective manual stipulation of partial bracketing and dependencies
- YY mode – external tokenization and tagging

Parser settings: unknown word handling

- Unknown open class words handled automatically:
  Beware the jubjub bird and shun the frumious bandersnatch.
- Default: ACE built-in POS tagger
- Alternate: call-out to TNT
  e.g. `ace -g erg/erg-1214.dat -1Tf --tnt-model=$LOGONROOT/coli/tnt/models/wsj`
- Performance empirically very similar
- YY mode – external tokenization and tagging

Parser settings: disambiguation models

- Maximum entropy model over derivation trees
- `ace -g erg/erg-1214.dat -1Tf --maxent=erg/wsj.mem`
- erg/ace/config.tdl: `maxent-model := "../redwoods.mem".`
- redwoods.mem – trained on all but WSJ
- wescience.mem – trained just on Wikipedia subset
- wsj.mem – trained just on WSJ

Efficiency vs. Precision in Parsing

- Parameters to control resource limits
  - **Time**: maximum number of seconds to use per sentence
    e.g. `ace ... --timeout=60`
  - **Memory**: maximum number of bytes to use for building the packed parse forest and for unpacking
    e.g. `ace ... --max-chart-megabytes=4000 --max-unpack-megabytes=6000`
  - **Number of analyses**: only unpack part of the forest
    e.g. `ace ... -1` or `ace ... -n 50`

Efficiency vs. Precision in Parsing (cont'd)

- 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]
- Specify probability threshold for discarding lexical items
  e.g. `ace ...  --ubertag=0.01`

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 predictably ill-formed, with correct semantics

Outline

More detailed analyses

- 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, Song, 2014]

Outline

Sample applications using ERS

- Scope of negation
- Logic to English (generation)
- Robot blocks world
- Database query via SQL

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}.

$$
\langle\, h_1, e_3,
$$

$$
\left|\begin{array}{l}
h_4\text{:\_the\_q}(x_6, h_7, h_5),\\
h_8\text{:named}(x_6, \textit{German}),\\
h_9\text{:\_send\_v\_for}(e_{10}, i_{11}, x_6),\\
h_9\text{:parg\_d}(e_{12}, e_{10}, x_6),\\
h_2\text{:\_but\_c}(e_3, h_9, e_{10}, h_{14}, e_{13}),\\
h_{14}\text{:\_profess\_v\_to}(e_{13}, x_6, h_{15}),\\
h_{16}\text{:\_know\_v\_1}(e_{17}, x_6, x_{18}),\\
h_{19}\text{:thing}(x_{18}),\\
h_{20}\text{:\_no\_q}(x_{18}, h_{21}, h_{22}),\\
h_{19}\text{:\_of\_p}(e_{23}, x_{18}, x_{24}),\\
h_{25}\text{:\_the\_q}(x_{24}, h_{27}, h_{26}),\\
h_{28}\text{:\_matter\_n\_of}(x_{24}, i_{29})
\end{array}\right|
$$

$$
\{\, h_{27} =_q h_{28},\ h_{21} =_q h_{19},\ h_{15} =_q h_{16},\ h_7 =_q h_8,\ h_1 =_q h_2 \,\}\,\rangle
$$

Approach

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

$$\langle h_1, e_3,$$
$$\left| \begin{array}{l} h_4\text{:\_the\_q}(x_6, h_7, h_5), \\ h_8\text{:named}(x_6, \textit{German}), \\ h_9\text{:\_send\_v\_for}(e_{10}, i_{11}, x_6), \\ h_9\text{:parg\_d}(e_{12}, e_{10}, x_6), \\ h_2\text{:\_but\_c}(e_3, h_9, e_{10}, h_{14}, e_{13}), \\ h_{14}\text{:\_profess\_v\_to}(e_{13}, x_6, h_{15}), \\ h_{16}\text{:\_know\_v\_1}(e_{17}, x_6, x_{18}), \\ h_{19}\text{:thing}(x_{18}), \\ h_{20}\text{:\_no\_q}(x_{18}, h_{21}, h_{22}), \\ h_{19}\text{:\_of\_p}(e_{23}, x_{18}, x_{24}), \\ h_{25}\text{:\_the\_q}(x_{24}, h_{27}, h_{26}), \\ h_{28}\text{:\_matter\_n\_of}(x_{24}, i_{29}) \end{array} \right|$$
$$\{ h_{27} =_q h_{28}, h_{21} =_q h_{19}, h_{15} =_q h_{16}, h_7 =_q h_8, h_1 =_q h_2 \} \rangle$$

Approach

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

$$
\begin{array}{l}
\langle\, h_1, e_3, \\
\quad \left|\begin{array}{l}
h_4\text{:\_the\_q}(x_6, h_7, h_5), \\
h_8\text{:named}(x_6, \textit{German}), \\
h_9\text{:\_send\_v\_for}(e_{10}, i_{11}, x_6), \\
h_9\text{:parg\_d}(e_{12}, e_{10}, x_6), \\
h_2\text{:\_but\_c}(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
h_{14}\text{:\_profess\_v\_to}(e_{13}, x_6, h_{15}), \\
h_{16}\text{:\_know\_v\_1}(e_{17}, x_6, x_{18}), \\
h_{19}\text{:thing}(x_{18}), \\
h_{20}\text{:\_no\_q}(x_{18}, h_{21}, h_{22}), \\
h_{19}\text{:\_of\_p}(e_{23}, x_{18}, x_{24}), \\
h_{25}\text{:\_the\_q}(x_{24}, h_{27}, h_{26}), \\
h_{28}\text{:\_matter\_n\_of}(x_{24}, i_{29})
\end{array}\right| \\
\quad \{\, h_{27} =_q h_{28}, h_{21} =_q h_{19}, h_{15} =_q h_{16}, h_7 =_q h_8, h_1 =_q h_2 \,\}\,\rangle
\end{array}
$$

Approach

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

$$
\begin{array}{l}
\langle\, h_1, e_3, \\
\quad h_4{:}\text{\_the\_q}(x_6, h_7, h_5), \\
\quad h_8{:}\text{named}(x_6, \textit{German}), \\
\quad h_9{:}\text{\_send\_v\_for}(e_{10}, i_{11}, x_6), \\
\quad h_9{:}\text{parg\_d}(e_{12}, e_{10}, x_6), \\
\quad h_2{:}\text{\_but\_c}(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
\quad h_{14}{:}\text{\_profess\_v\_to}(e_{13}, x_6, h_{15}), \\
\quad h_{16}{:}\text{\_know\_v\_1}(e_{17}, x_6, x_{18}), \\
\quad h_{19}{:}\text{thing}(x_{18}), \\
\quad h_{20}{:}\text{\_no\_q}(x_{18}, h_{21}, h_{22}), \\
\quad h_{19}{:}\text{\_of\_p}(e_{23}, x_{18}, x_{24}), \\
\quad h_{25}{:}\text{\_the\_q}(x_{24}, h_{27}, h_{26}), \\
\quad h_{28}{:}\text{\_matter\_n\_of}(x_{24}, i_{29}) \\
\{\, h_{27} =_q h_{28}, h_{21} =_q h_{19}, h_{15} =_q h_{16}, h_7 =_q h_8, h_1 =_q h_2 \,\}\,\rangle
\end{array}
$$

Approach

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

$$
\langle h_1, e_3,
$$

$$
\begin{array}{|l|}
h_4\text{:\_the\_q}(x_6, h_7, h_5), \\
h_8\text{:named}(x_6, \textit{German}), \\
h_9\text{:\_send\_v\_for}(e_{10}, i_{11}, x_6), \\
h_9\text{:parg\_d}(e_{12}, e_{10}, x_6), \\
h_2\text{:\_but\_c}(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
h_{14}\text{:\_profess\_v\_to}(e_{13}, x_6, h_{15}), \\
h_{16}\text{:\_know\_v\_1}(e_{17}, x_6, x_{18}), \\
h_{19}\text{:thing}(x_{18}), \\
h_{20}\text{:\_no\_q}(x_{18}, h_{21}, h_{22}), \\
h_{19}\text{:\_of\_p}(e_{23}, x_{18}, x_{24}), \\
h_{25}\text{:\_the\_q}(x_{24}, h_{27}, h_{26}), \\
h_{28}\text{:\_matter\_n\_of}(x_{24}, i_{29})
\end{array}
$$

$$
\{ h_{27} =_q h_{28}, h_{21} =_q h_{19}, h_{15} =_q h_{16}, h_7 =_q h_8, h_1 =_q h_2 \} \rangle
$$

Approach

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

$$
\langle\, h_1, e_3,\ \left|\begin{array}{l}
h_4{:}\_the\_q(x_6, h_7, h_5),\\
h_8{:}named(x_6, German),\\
h_9{:}\_send\_v\_for(e_{10}, i_{11}, x_6),\\
h_9{:}parg\_d(e_{12}, e_{10}, x_6),\\
h_2{:}\_but\_c(e_3, h_9, e_{10}, h_{14}, e_{13}),\\
h_{14}{:}\_profess\_v\_to(e_{13}, x_6, h_{15}),\\
h_{16}{:}\_know\_v\_1(e_{17}, x_6, x_{18}),\\
h_{19}{:}thing(x_{18}),\\
h_{20}{:}\_no\_q(x_{18}, h_{21}, h_{22}),\\
h_{19}{:}\_of\_p(e_{23}, x_{18}, x_{24}),\\
h_{25}{:}\_the\_q(x_{24}, h_{27}, h_{26}),\\
h_{28}{:}\_matter\_n\_of(x_{24}, i_{29})
\end{array}\right|
\{\, h_{27} =_q h_{28}, h_{21} =_q h_{19}, h_{15} =_q h_{16}, h_7 =_q h_8, h_1 =_q h_2 \,\}\,\rangle
$$

Approach

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

$$
\begin{array}{l}
\langle\, h_1, e_3, \\
\quad h_4\text{:\_the\_q}\langle 0{:}3\rangle(x_6, h_7, h_5), \\
\quad h_8\text{:named}\langle 4{:}10\rangle(x_6, \mathit{German}), \\
\quad h_9\text{:\_send\_v\_for}\langle 15{:}19\rangle(e_{10}, i_{11}, x_6), \\
\quad h_9\text{:parg\_d}\langle 15{:}19\rangle(e_{12}, e_{10}, x_6), \\
\quad h_2\text{:\_but\_c}\langle 24{:}27\rangle(e_3, h_9, e_{10}, h_{14}, e_{13}), \\
\quad h_{14}\text{:\_profess\_v\_to}\langle 28{:}37\rangle(e_{13}, x_6, h_{15}), \\
\quad h_{16}\text{:\_know\_v\_1}\langle 41{:}45\rangle(e_{17}, x_6, x_{18}), \\
\quad h_{19}\text{:thing}\langle 46{:}53\rangle(x_{18}), \\
\quad h_{20}\text{:\_no\_q}\langle 46{:}53\rangle(x_{18}, h_{21}, h_{22}), \\
\quad h_{19}\text{:\_of\_p}\langle 54{:}56\rangle(e_{23}, x_{18}, x_{24}), \\
\quad h_{25}\text{:\_the\_q}\langle 57{:}60\rangle(x_{24}, h_{27}, h_{26}), \\
\quad h_{28}\text{:\_matter\_n\_of}\langle 61{:}68\rangle(x_{24}, i_{29}) \\
\{\, h_{27} =_q h_{28}, h_{21} =_q h_{19}, h_{15} =_q h_{16}, h_7 =_q h_8, h_1 =_q h_2 \,\}\,\rangle
\end{array}
$$

Results

- As of 2014, state of the art for this task

	Scopes			Tokens		
Method	**Prec**	**Rec**	**F_1**	**Prec**	**Rec**	**F_1**
Read et al 2012	87.4	61.5	72.2	82.0	**88.8**	85.3
ERS Crawler	**87.8**	43.4	58.1	78.8	66.7	72.2
Combined System	87.6	**62.7**	**73.1**	**82.6**	88.5	**85.4**

Data/software for reproducibility:
`http://www.delph-in.net/crawler/`

Task: Generate English from First-Order Logic

- Online course on introductory logic
  Textbook: Barker-Plummer, Barwise and Etchemendy, *Language, Proof, and Logic, 2nd Edition*
- 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:

```
large(a)&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
  INDEX: e13 [ e SORT: collective SF: prop TENSE: pres PERF: - ]
  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

`large(a)&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

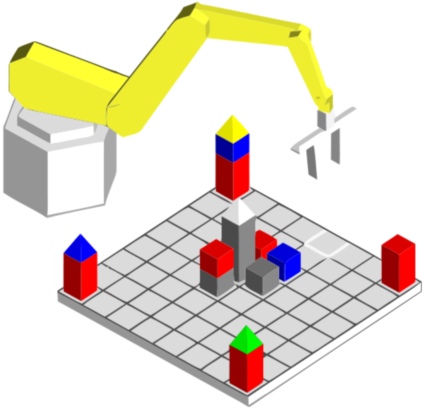
```
(cube(a)&cube(b))->leftof(a,b)
```

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, 2014b]

Game board illustration



Example of robot command

Pick up the turquoise pyramid standing over a white cube

$$\langle\, h_0, e_2,$$
$$\left|\begin{array}{l} h_4\text{:pronoun\_q}(x_3, h_5, h_6), \\ h_7\text{:pron}(x_3), \\ h_1\text{:\_pick\_v\_up}(e_2, x_3, x_8), \\ h_9\text{:\_the\_q}(x_8, h_{10}, h_{11}), \\ h_{12}\text{:\_turquoise\_a\_1}(e_{13}, x_8), \\ h_{12}\text{:\_pyramid\_n\_1}(x_8), \\ h_{12}\text{:\_stand\_v\_1}(e_{14}, x_8), \\ h_{12}\text{:\_over\_p}(e_{15}, e_{14}, x_{16}), \\ h_{17}\text{:\_a\_q}(x_{16}, h_{18}, h_{19}), \\ h_{20}\text{:\_white\_a\_1}(e_{21}, x_{16}), \\ h_{20}\text{:\_cube\_n\_1}(x_{16}) \end{array}\right|$$
$$\{\, h_{18} =_q h_{20}, h_{10} =_q h_{12}, h_5 =_q h_7, h_0 =_q h_1 \,\}\,\rangle$$

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))
```

ERS to RCL mechanism

- Traverse ERS graph starting from TOP/INDEX
- Top-level `_v_` predicates become

  `(event: (action: P) (entity: ARG1))`

  where action P is determined by predicate name
- `_n_` predicates become `(entity: (type: P))` where type P is determined by predicate name
- Remaining predicates become `(color:)` and `(spatial-relation:)` decorations

Database query via SQL

- Demo evaluating statements about a small world of shapes
- Mapping from ERS to SQL
- Study of interactions of quantifiers and other scopal operators
- Website for experimentation:
  `sweaglesw.org/linguistics/objects-demo/`
- Sample statements:
  Most squares are red.
  Some blue triangles are not above a square.

Acknowledgements

We are grateful to Ann Copestake and Stephan Oepen for their considerable help in preparing these materials.

References I

Banarescu, L., Bonial, C., Cai, S., Georgescu, M., Griffitt, K., Hermjakob, U., Knight, K., Koehn, P., Palmer, M., and Schneider, N. (2013).
Abstract meaning representation for sembanking.
In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, page 178 – 186, Sofia, Bulgaria.

Bond, F., Oepen, S., Nichols, E., Flickinger, D., Velldal, E., and Haugereid, P. (2011).
Deep open-source machine translation.
Machine Translation, 25:87–105.

Callmeier, U. (2002).
Preprocessing and encoding techniques in PET.
In Oepen, S., Flickinger, D., Tsujii, J., and Uszkoreit, H., editors, *Collaborative Language Engineering. A Case Study in Efficient Grammar-based Processing*, page 127 – 140. CSLI Publications, Stanford, CA.

Carter, D. (1997).
The TreeBanker. A tool for supervised training of parsed corpora.
In *Proceedings of the Workshop on Computational Environments for Grammar Development and Linguistic Engineering*, page 9 – 15, Madrid, Spain.

References II

Copestake, A. (2002).
Implementing Typed Feature Structure Grammars.
CSLI Lecture Notes. Center for the Study of Language and Information, Stanford,California.

Copestake, A. (2009).
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.

Copestake, A., Emerson, G., Goodman, M. W., Horvat, M., Kuhnle, A., and Muszyńska, E. (2016).
Resources for building applications with dependency minimal recursion semantics.
In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, Portorož, Slovenia.

Copestake, A., Flickinger, D., Pollard, C., and Sag, I. A. (2005).
Minimal Recursion Semantics. An introduction.
Research on Language and Computation, 3(4):281 – 332.

References III

Dridan, R. (2013).
Ubertagging. Joint segmentation and supertagging for English.
In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Seattle, WA, USA.

Flickinger, D. (2000).
On building a more efficient grammar by exploiting types.
Natural Language Engineering, 6 (1):15 – 28.

Flickinger, D. (2011).
Accuracy vs. robustness in grammar engineering.
In Bender, E. M. and Arnold, J. E., editors, *Language from a Cognitive Perspective: Grammar, Usage, and Processing*, page 31 – 50. Stanford: CSLI Publications.

Herbelot, A. (2013).
What is in a text, what isn't, and what this has to do with lexical semantics.
In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013)*, pages 321–327, Potsdam, Germany.

Herbelot, A. and Copestake, A. (2006).
Acquiring Ontological Relationships from Wikipedia Using RMRS.
In *Proceedings of the ISWC 2006 Workshop on Web Content.*

References IV

Ivanova, A., Oepen, S., Øvrelid, L., and Flickinger, D. (2012).
Who did what to whom? A contrastive study of syntacto-semantic dependencies.
In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 2–11, Jeju, Republic of Korea.

Kouylekov, M. and Oepen, S. (2014).
Semantic technologies for querying linguistic annotations. An experiment focusing on graph-structured data.
In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, page 4331 – 4336, Reykjavik, Iceland.

Lien, E. and Kouylekov, M. (2014).
Entailment recognition using Minimal Recursion Semantics.
In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014*, pages 699–703.

MacKinlay, A., Martinez, D., and Baldwin, T. (2009).
Biomedical Event Annotation with CRFs and Precision Grammars.
In *Proceedings of the BioNLP 2009 Workshop Companion Volume for Shared Task*, pages 77–85.

References V

Morante, R. and Blanco, E. (2012).
*SEM 2012 shared task: Resolving the scope and focus of negation.
In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*, page 265 – 274, Montréal, Canada.

Nichols, E., Bond, F., and Flickinger, D. (2005).
Robust Ontology Acquisition from Machine-Readable Dictionaries.
In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1111–1116, Edinburgh.

Oepen, S., Flickinger, D., Toutanova, K., and Manning, C. D. (2002).
Lingo Redwoods. A rich and dynamic treebank for HPSG.
In *Proceedings of the 1st International Workshop on Treebanks and Linguistic Theories*, page 139 – 149, Sozopol, Bulgaria.

Oepen, S., Flickinger, D., Toutanova, K., and Manning, C. D. (2004).
LinGO Redwoods. A rich and dynamic treebank for HPSG.
Research on Language and Computation, 2(4):575 – 596.

Oepen, S. and Lønning, J. T. (2006).
Discriminant-based MRS banking.
In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, page 1250 – 1255, Genoa, Italy.

References VI

Packard, W. (2014a).
UW-MRS: Leveraging a deep grammar for robotic spatial commands.
In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 812–816, Dublin, Ireland.

Packard, W. (2014b).
UW-MRS: Leveraging a deep grammar for robotic spatial commands.
In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, page 812 – 816, Dublin, Ireland.

Packard, W., Bender, E. M., Read, J., Oepen, S., and Dridan, R. (2014).
Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem.
In *Proceedings of the 52nd Meeting of the Association for Computational Linguistics*, pages 69–78, Baltimore, Maryland.

Read, J., Velldal, E., Øvrelid, L., and Oepen, S. (2012).
UiO1: constituent-based discriminative ranking for negation resolution.
In *Proceedings of the 1st Joint Conference on Lexical and Computational Semantics*, page 310 – 318, Montréal, Canada.

References VII

Song, S. (2014).
A Grammar Library for Information Structure.
PhD thesis, University of Washington.

Song, S. and Bender, E. M. (2012).
Individual constraints for information structure.
In *Proceedings of the 19th International Conference on Head-Driven Phrase Structure Grammar (HPSG 2012)*, page 330 – 348, Daejeon, Korea.

Toutanova, K., Manning, C. D., Flickinger, D., and Oepen, S. (2005).
Stochastic HPSG Parse Disambiguation using the Redwoods Corpus.
Research on Language and Computation, 3:83 – 105.

Yao, X., Bouma, G., and Zhang, Y. (2012).
Semantics-based question generation and implementation.
Dialogue & Discourse, 3(2):11–42.

Comparison with (enhanced) universal dependencies

Cats are easy to please.

```
nsubj(easy-3, cats-1)      nsubj(please-5, cats-1)
cop(easy-3, are-2)         root(ROOT-0, easy-3)
mark(please-5, to-4)       xcomp(easy-3, please-5)
```

from online demo at `nlp.stanford.edu`

ERS:

$$\langle h_1, e_3, \left| \begin{array}{l} h_4\text{:udef\_q}(x_6, h_5, h_7), \\ h_8\text{:\_cat\_n\_1}(x_6), \\ h_2\text{:\_easy\_a\_for}(e_3, h_9, i_{10}), \\ h_{11}\text{:\_please\_v\_1}(e_{12}, i_{10}, x_6) \end{array} \right| \{ h_1 =_q h_2, h_5 =_q h_8, h_9 =_q h_{11} \} \rangle$$

Comparison with universal dependencies (cont.)

It is easy to please cats

```
nsubj(easy-3, It-1)         nsubj(please-5, It-1)
cop(easy-3, is-2)           root(ROOT-0, easy-3)
mark(please-5, to-4)        xcomp(easy-3, please-5)
dobj(please-5, cats-6)
```

Cats are easy to please.

```
nsubj(easy-3, cats-1)       nsubj(please-5, cats-1)
cop(easy-3, are-2)          root(ROOT-0, easy-3)
mark(please-5, to-4)        xcomp(easy-3, please-5)
```

*MRS is the same for both sentences.

Comparison with AMR [Banarescu et al., 2013]

Cats are easy to please.
It is easy to please cats.

According to the AMR manual:

```
(e / easy
   :domain (p / please-01
                 :ARG1 (c / cat )))
```

ERS:

$$\langle h_1, e_3, \left| \begin{array}{l} h_4\text{:udef\_q}(x_6, h_5, h_7), \\ h_8\text{:\_cat\_n\_1}(x_6), \\ h_2\text{:\_easy\_a\_for}(e_3, h_9, i_{10}), \\ h_{11}\text{:\_please\_v\_1}(e_{12}, i_{10}, x_6) \end{array} \right| \{ h_1 =_q h_2, h_5 =_q h_8, h_9 =_q h_{11} \} \rangle$$

Comparison with AMR

Cats are easy to please. | It is easy to please cats.
Pleasing cats is easy.

According to AMR manual, all should have structure:

```
(e / easy
   :domain (p / please-01
                :ARG1 (c / cat )))
```

ERS for *Pleasing cats is easy.*:

$$\langle h_1, e_3, \left| \begin{array}{l} h_4\text{:udef\_q}(x_6, h_5, h_7), \\ h_8\text{:nominalization}(x_6, h_9), \\ h_9\text{:\_please\_v\_1}(e_{10}, i_{12}, x_{11}), \\ h_{13}\text{:udef\_q}(x_{11}, h_{14}, h_{15}), \\ h_{16}\text{:\_cat\_n\_1}(x_{11}), \\ h_2\text{:\_easy\_a\_for}(e_3, x_6, i_{17}) \end{array} \right| \{ h_1 =_q h_2, h_5 =_q h_8, h_{14} =_q h_{16} \} \rangle$$