Implementing HPSG

Why implement grammars?

- Practical applications:
  - Machine translation (symbolic, hybrid)
  - Automated customer service response
  - Speech prostheses
  - Hybrid Q&A systems (DeepThought)
  - Language tutors

Why implement grammars?

- Linguistic hypothesis testing:
  - Particular analyses
  - Interaction of analyses
  - Against test suites or corpora
  - Crosslinguistic testing of formalism choices
  - Crosslinguistic testing of proposed universals

Bottom Line

- People are better than computers at language
- BUT: Computers are better than people at keeping track of the details of formal analyses of language
- Further: The precision and coverage required by implementation keeps theoretical work grounded in actual data.

Resources: Delph-In

- LKB: grammar development environment (including parser and generator)
- PET: industrial strength parser
- [incr tsdb()]: Competence and performance profiling laboratory (regression testing)
- Redwoods: Software for creating dynamic, grammar-associated treebanks

Parsing Complexity

- Depends on:
  - Formalism
  - Grammar
  - Sentence length
  - Lexical ambiguity
  - Best first v. exhaustive

How fast is the fast parser?

Set	Aggregate	total items ♯	word string ϕ	lexical entries ϕ	total results ♯	parser analyses ϕ	passive edges ϕ	fs size ϕ
'tsnlp'	wellformed	1574	4·96	13·8	945	2·00	86	273
	illformed	2775	4·50	11·5	409	1·83	39	257
'csli'	wellformed	918	6·45	15·3	732	2·16	115	302
	illformed	375	6·11	14·9	85	2·31	84	298
'aged'	wellformed	96	8·41	23·1	72	7·00	292	315
'blend'	wellformed	1910	11·13	32·1	1008	51·39	1181	336
	illformed	142	11·05	34·2	24	20·33	611	317

Table 2: Reference data sets used in comparison and benchmarking with the LinGO grammar.

Version	Platform	Test Set	filter %	etasks ϕ	pedges ϕ	tcpu ϕ (s)	space ϕ (kb)
October 1996	PAGE	*'tsnlp'*	49·9	656	44	3·69	19016
		'aged'	51·3	1763	97	21·16	79093
August 1999	PET (cheap)	*'tsnlp'*	93·9	170	55	0·03	333
		'aged'	95·1	753	292	0·14	1435
		'blend'	95·5	3084	1140	0·65	10589

(generated by [incr tsdb()] at 5-nov-1999 (21:23 h)

Table 6: Development of salient performance parameters (PAGE vs. PET) over three years.

Resources: Delph-In

- Resource grammars: English, German, Japanese, Norwegian, Modern Greek...
- The Grammar Matrix: Rapid-prototyping of scaleable precision grammars
- And more: Delph-In: www.delph-in.net

Related Work

- TRALE: Another grammar development environment for HPSG
- ParGram: Parallel Grammar development for multiple languages in LFG

Grammar Matrix: Motivation

- ERG stats (5/2005)
  - 140,000 lines of code (25,000 exclusive of lexicon)
  - ~3000 types
  - 16+ person-years of effort
- How much of that is useful in other languages?
- How much faster can we develop the next grammar?

Grammar Matrix: Motivation

- Promote consistent semantic representations
  - Reuse ‘downstream’ technology in NLU/ NLG applications
  - Transfer-based MT (symbolic or stochastic)
- Crosslinguistic hypothesis testing
- Endangered language documentation

Matrix: Proposed Universals

- Words and phrases combine to make larger phrases.
- The semantics of a phrase is determined by the meaning of its parts and how they're put together.
- Some rules for phrases add semantics.
- No rule can remove semantic information.

Matrix: Proposed Universals

- Most phrases have a head daughter.
- Heads determine the types of arguments they require, and how they combine semantically with those arguments.
- Modifiers determine the types of heads they modify, and how they combine semantically with those heads.

Matrix: Proposed Universals

- All NPs are associated semantically with a quantifier.
- Quantification over events is different from quantification over individuals.
- Definiteness/discourse status is expressed in the same way as distinctions between quantifiers.

Matrix: Modules

- Plenty of patterns recur without being universal.
- Current work: Development of ‘modules’ representing different strategies for: word order, sentential negation, coordination, question formation...
- Challenges: Modules aren’t all that ‘modular’

Example: Word order

- The present word order modules constrain:
  - Order of daughters in head-subject and head-complement rules
  - Order of application of head-subject and head-complement rules
- Potentially interacts with:
  - Clause type, part of speech type (of head or dependent), finiteness, ...

Matrix Modules

- Each piece carefully designed to account for one particular aspect of a language...
- ... while interacting appropriately with other modules.
- How best to organize/maintain the various pieces in such a resource?
- How best to ask linguists for the information which will guide module choices?

Language

e.g., Basque or Hawaiian_Creole:

Word Order

Please indicate which pattern best describes the basic word order of your language:

- a) SOV
- b) SVO
- c) VSO
- d) OSV
- e) OVS
- f) VOS
- g) V-final
- h) V-initial
- i) free (pragmatically determined) word order

Note: Modules for V2 order (auxiliary second, all else free or finite verb second, non-finite verb clause-finally) and differing word order between matrix and subordinate clauses are currently under development.

Does your language have determiners (as indpendent words)? yes no

If so, what is the order of determiners with respect to nouns? Noun-Det Det-Noun

---

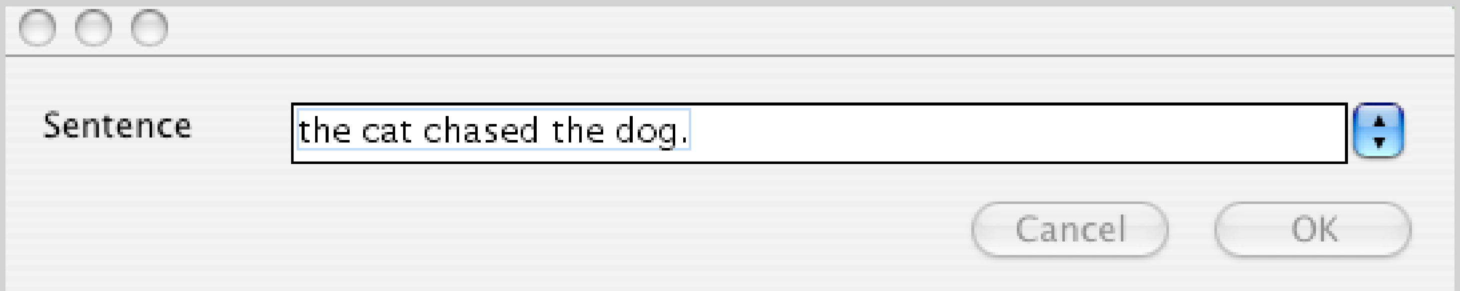
Basic Lexicon

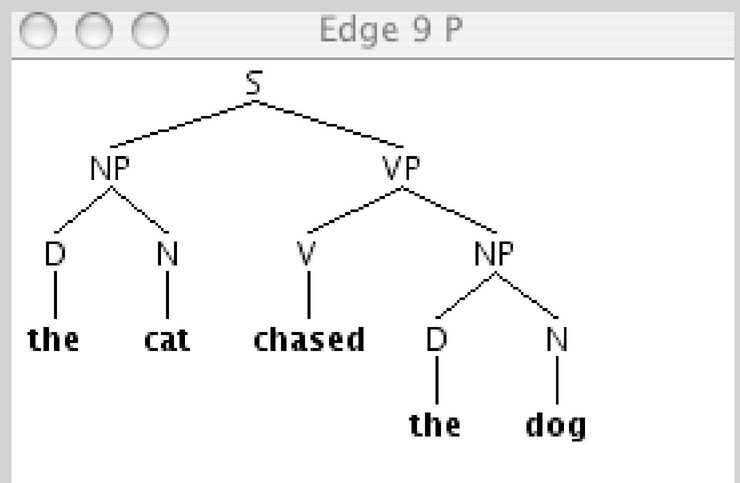
Please use full form lexical entries (not stem forms or citation forms). This form can only accept lower ascii alphanumerics (a-z, A-Z, 0-9, _).

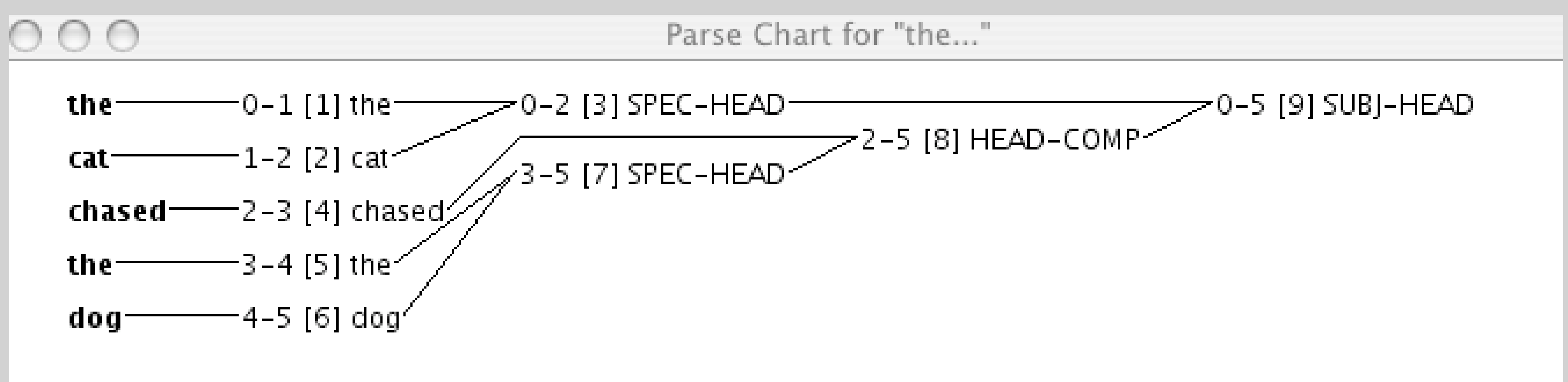
Noun 1:

- Spelling:
- Predicate name (e.g., _cat_n_rel):
- For this noun, a specifier (determiner) is obligatory optional impossible

Working with the LKB







Lab: Call for Participation

- Visit the course website for lab instructions
- Software is available on the PCs on the computer classroom, or for download to your own computer
- I'll be checking the bulletin board frequently!