The Lexicon in the LinGO Grammar Matrix: Cross-linguistic Hypotheses about Words

Emily M. Bender

February 24, 2004

Overview

- Precision grammars
- What is the Matrix?
- Phrase structure rules and the syntax-semantics interface in the Matrix
- Lexical types in the Matrix
- Future work: Montage

Why precision grammars?

- Applications requiring natural language understanding: automated email response, CALL, dialogue systems, precision machine translation
- Improved resources as input to machine learning techniques
- Linguistic and cross-linguistic hypothesis testing, language documentation

Resources

- HPSG (Pollard & Sag 1994)
- MRS (Copestake et al 2003)
- LKB (Copestake 2002)
- PET/cheap (Callmeier 2000, Oepen et al 2002a)
- Treebanking/stochastic parse selection techniques (Oepen et al 2002b, Toutanova et al 2002)
- DeepThought: Interfaces to robust shallow parsing (www.eurice.de/deepthought)

Challenges

- Labor-intensive, expensive process
- Interoperability across grammars
- Reusability of resources/analyses
- Robustness in the face of real-world language use

Precision grammar development

- Desiderata:
  - Core/periphery compatibility
  - Scalability
  - Maintainability
  - Semantic compositionality
  - Standardized semantic output

The LinGO Grammar Matrix

- Cross-linguistically valid core grammar:
  - rapid initial start-up
  - steady expansion to broad coverage
- Standardized semantic output:
  - interoperability
- HPSG/Construction Grammar formalism:
  - scalability
  - modularity
  - core and periphery

(Bender, Flickinger & Oepen 2002)

HPSG and types

- HPSGs are typed feature structure grammars
- Grammatical information represented as constraints on possible words and phrases
- Constraints are stated on types, representing classes of linguistic objects
- Types are organized into a multiple-inheritance hierarchy, representing
  - Generalizations at varying granularities
  - Cross-cutting generalizations

The LinGO Grammar Matrix

- Constraints extracted from (experience with) large-scale implemented HPSGs:
  - English
  - German
  - Japanese
- Preliminary version used for:
  - Norwegian
  - Italian
  - Modern Greek

What is in the Matrix (v0.6)?

- Types defining basic feature geometry
- Underspecified construction types
- Implementation of compositional semantics
- Definitions of semantic structures
- Collateral files for interaction with the LKB grammar development environment (Copestake 2002).
- *New*: Initial hypotheses about lexical types

How big is the Matrix?

Grammar	Types	Lines of code
Matrix v0.6	202	1208
Matrix with lexical types	245	1726
ERG (12/03)	3413	22518
Modern Greek (2/04)	816	4331
NorSource (12/03)	2077	6415

What will be in the Matrix?

- Universal aspects of a Sem-I (semantics interface)
- Modules for non-universal yet recurring phenomena, e.g.:
  - tense/aspect systems
  - numeral classifiers
- Support for creating test suites for regression testing
- ...

Early experiments: Modern Greek

- In development since 1/03; primary developer works just 10 hours a week
- Phenomena covered:
  - Internal syntax of NPs
  - Subordinate clauses, incl relative clauses
  - Long-distance dependencies
  - Raising and control
  - Politeness constructions
  - Cliticization
  - Valence alternations
  - Word order phenomena

(Kordoni & Neu 2003a,b)

Early experiments: Scandinavian Matrix

- Norwegian grammar underdevelopment since 1/02 (Matrix v0.1).
- Extensive coverage, including: linking, predicative complements, presentational constructions, passive, light pronouns...
- Using it as a basis for Swedish and Danish grammars, possibly directly, possibly as the basis of a Scandinavian Matrix

(Hellan & Haugereid 2003)

Hypotheses in the Matrix

- Essentially a bottom-up approach to UG
- First pass hypotheses concern which parts of existing grammars are likely to be cross-linguistically useful
- Study Matrix-derived grammars for a variety of languages to see what is and isn't useful
- Move some constraints/subsystems into separate modules
- Look for exhaustive classifications on the basis of actual grammars

Sample phrase structure rule: head-complement cxs

$$\begin{bmatrix}\text{COMPS } \boxed{2}\end{bmatrix} \rightarrow \begin{bmatrix}\text{COMPS } \langle \boxed{1} \rangle \oplus \boxed{2}\end{bmatrix}, \begin{bmatrix}\text{SYNSEM } \boxed{1}\end{bmatrix}$$

- Make a phrase out of a word or phrase looking for some complements and it's first complement.
- Unify the *synsem* of the complement with the complement requirement of the head.
- Gather up the semantic contributions of both daughters.
- Collect non-local features from the head daughter.

Sample phrase structure rule: head-complement cxs

```
basic-head-comp-phrase := head-valence-phrase & head-compositional &
        binary-headed-phrase &
  [ SYNSEM canonical-synsem &
     [ LOCAL.CAT [ MC #mc,
                       VAL [ SUBJ #subj,
                             COMPS #comps,
                             SPR #spr ],
                       POSTHEAD #ph ],
           LEX #lex ],
    HEAD-DTR.SYNSEM [ LOCAL.CAT [ MC #mc,
                                  VAL [ SUBJ #subj,
                                        COMPS < #synsem . #comps >,
                                        SPR #spr ],
                                  POSTHEAD #ph ],
                      LEX #lex ],
    NON-HEAD-DTR.SYNSEM #synsem & canonical-synsem,
    C-CONT [ RELS <! !>,
       HCONS <! !> ] ].
```

Long distance dependencies

- Topicalization (SLASH), pied-piping in relative clauses (REL) and questions (QUE)
- SLASH: Each phrase records via this feature whether there is anything 'missing' inside it
- Certain constructions require a daughter with a missing element
- Heads collect non-local feature values of their dependents, and pass them up to their mothers, except in head-filler constructions
- Traceless analysis

(Bouma et al 2001)

Syntax-semantics interface

- MRS: Flat semantic representations; underspecification of scope.
- With scope resolved, equivalent to predicate logic.
- The heart of an MRS is a bag of relations.
- Every constituent exposes a small amount of information about its relations via the HOOK:
  - A distinguished index
  - The topmost handle (for scope purposes)
  - The index of its external argument (if any)

(Copestake et al 2003, Flickinger & Bender 2003)

Convergent semantic representations

- MRS designed for: expressivity, computational tractability, scalability, underspecification
- The Matrix aids grammar engineers in producing valid MRS representations
- The Matrix also helps standardize representations of specific linguistic phenomena, e.g., number names, nominalizations, etc.
- Standardized output ensures interoperability

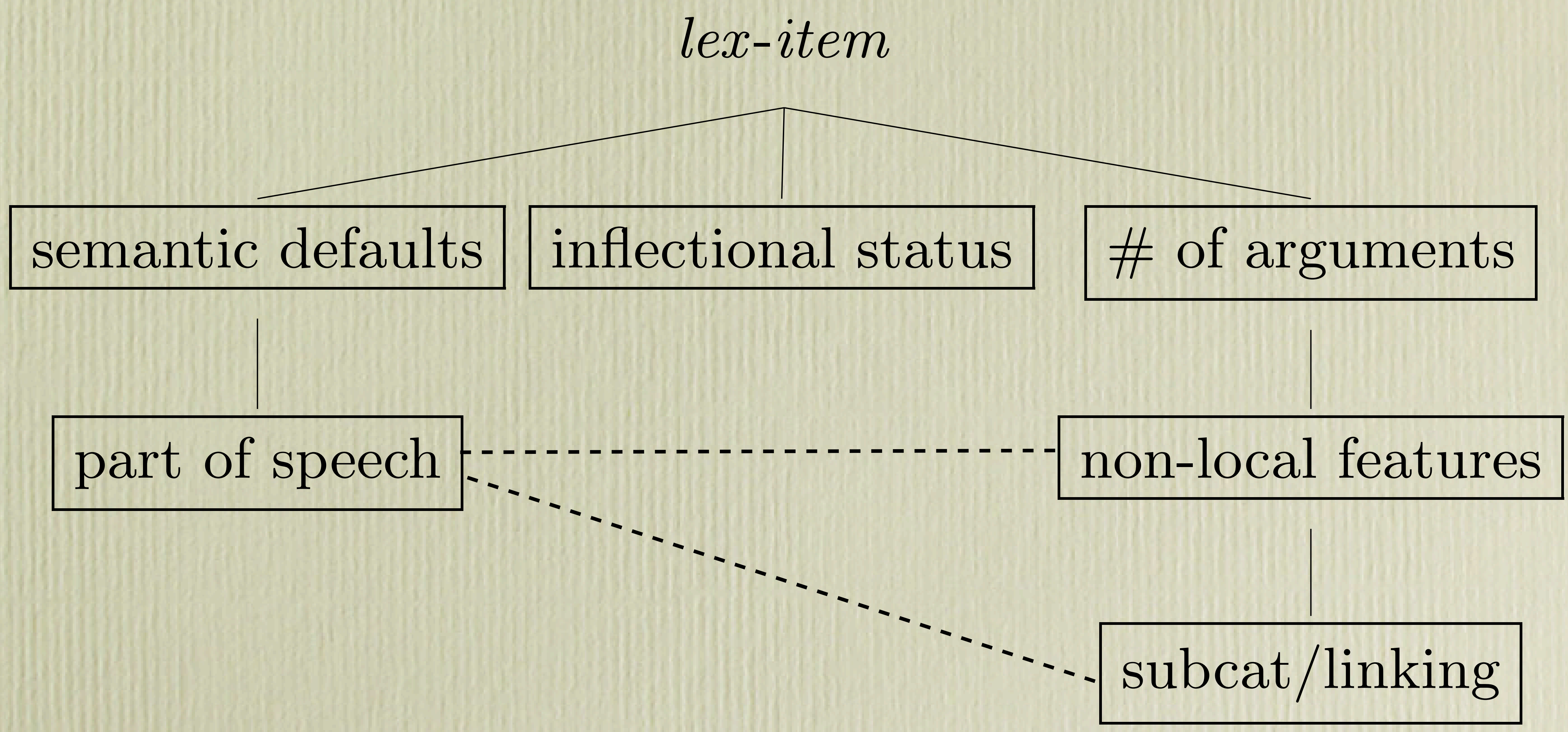
Lexical types: Desiderata

- Illustrate properties of words required by the specific encoding of the phrase structure rules
- Standardize lexical aspects of the syntax-semantics interface
- Encode a space of possibilities that are likely to be useful cross-linguistically
- Allow for extensibility without changing the existing Matrix hierarchy

Lexical types: Dimensions of classification

- Semantic contribution
- Inflectional status; light vs. heavy status
- Number of arguments
- Introduction and amalgamation of non-local features (SLASH, REL, QUE)
- Subcategorization/linking
- Part of speech

Lexical types: Dimensions of classification



Semantic defaults

- *norm-hook-lex-item*: the HOOK features are related in the ordinary way to the main semantic relation (KEY relation)
- *single-rel-lex-item*: lexical item contributes exactly one relation
- *no-hcons-lex-item*: lexical item contributes no handle constraints
- *norm-sem-lex-item*: all of the above

Semantic defaults

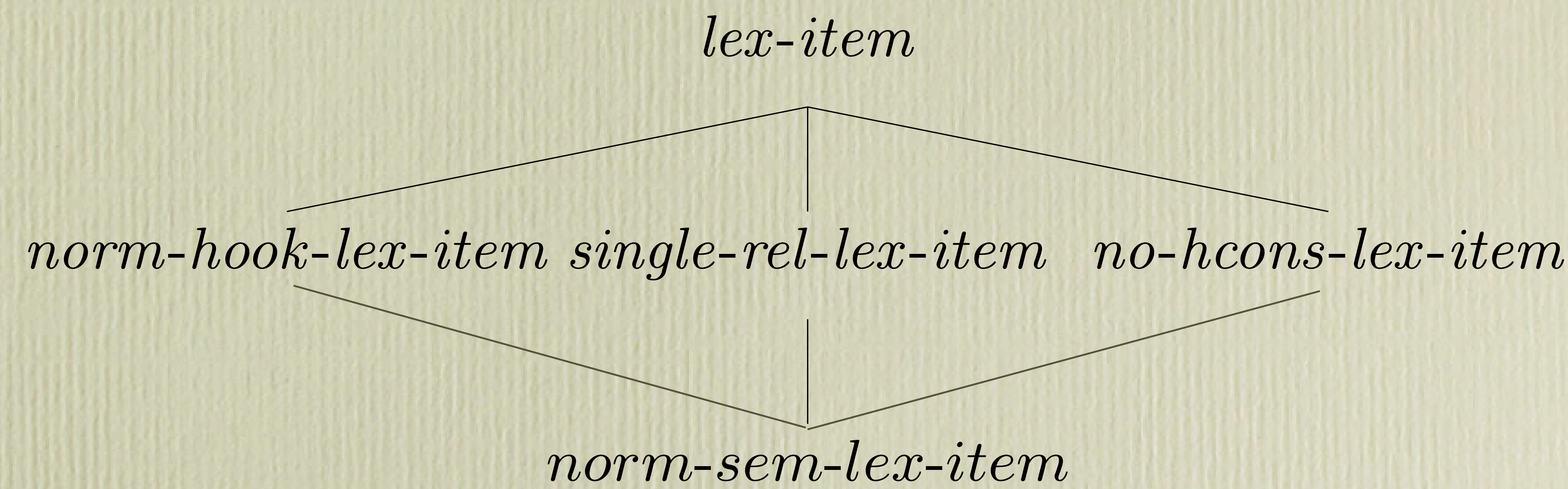
```
norm-hook-lex-item := lex-item &
  [ SYNSEM [ LOCAL.CONT [ HOOK [ LTOP #ltop,
                                 INDEX #index ],
                          RELS.LIST.FIRST #keyrel ],
             LKEYS.KEYREL #keyrel & [ LBL #ltop,
                                      ARG0 #index ] ] ].

single-rel-lex-item := lex-item &
  [ SYNSEM.LOCAL.CONT.RELS 1-dlist ].

no-hcons-lex-item := lex-item &
  [ SYNSEM.LOCAL.CONT.HCONS 0-dlist ].

norm-sem-lex-item := norm-hook-lex-item &
                     single-rel-lex-item &
                     no-hcons-lex-item.
```

Semantic defaults



Inflectional status

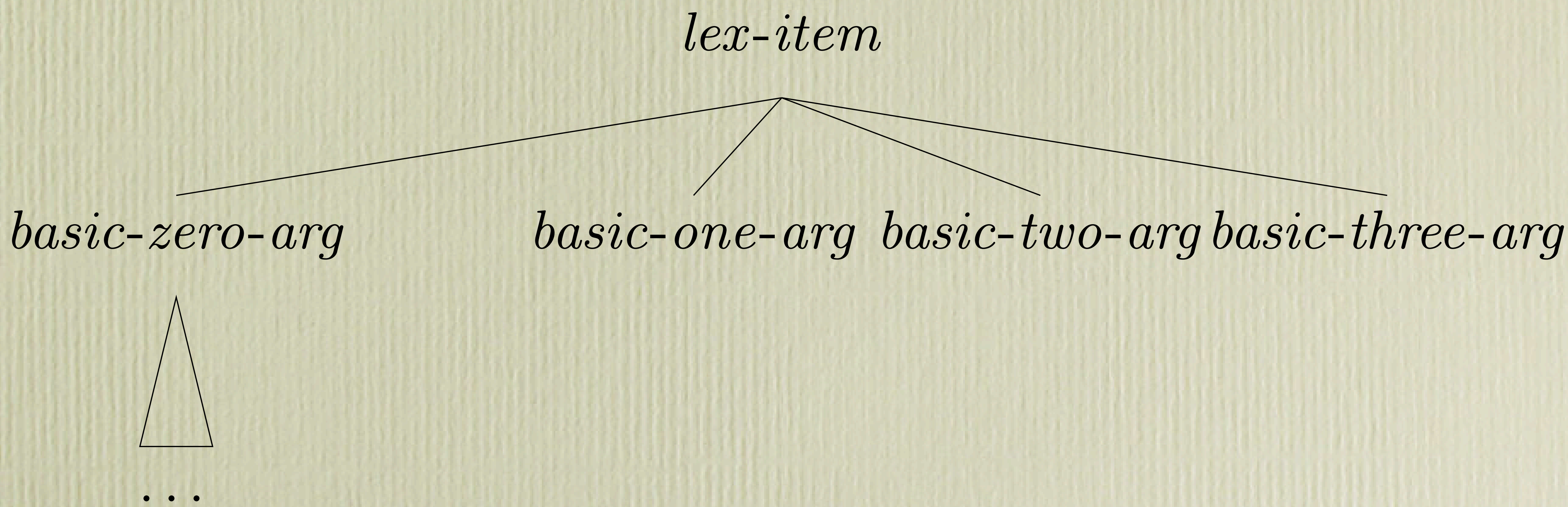
lex-item

$$\begin{bmatrix} \textit{uninflected-lexeme} & \\ \text{INFLECTED} & - \end{bmatrix} \quad \begin{bmatrix} \textit{fully-inflected-lexeme} & \\ \text{INFLECTED} & + \end{bmatrix}$$

Number of arguments/ non-local features

- How many arguments does the lexical item select for (subject, complements, specifier)?
- Does it amalgamate non-local features from all of the dependents?
- Does it introduce non-local features of it's own?
- We expect more types in this dimension will need to be added for particular languages.

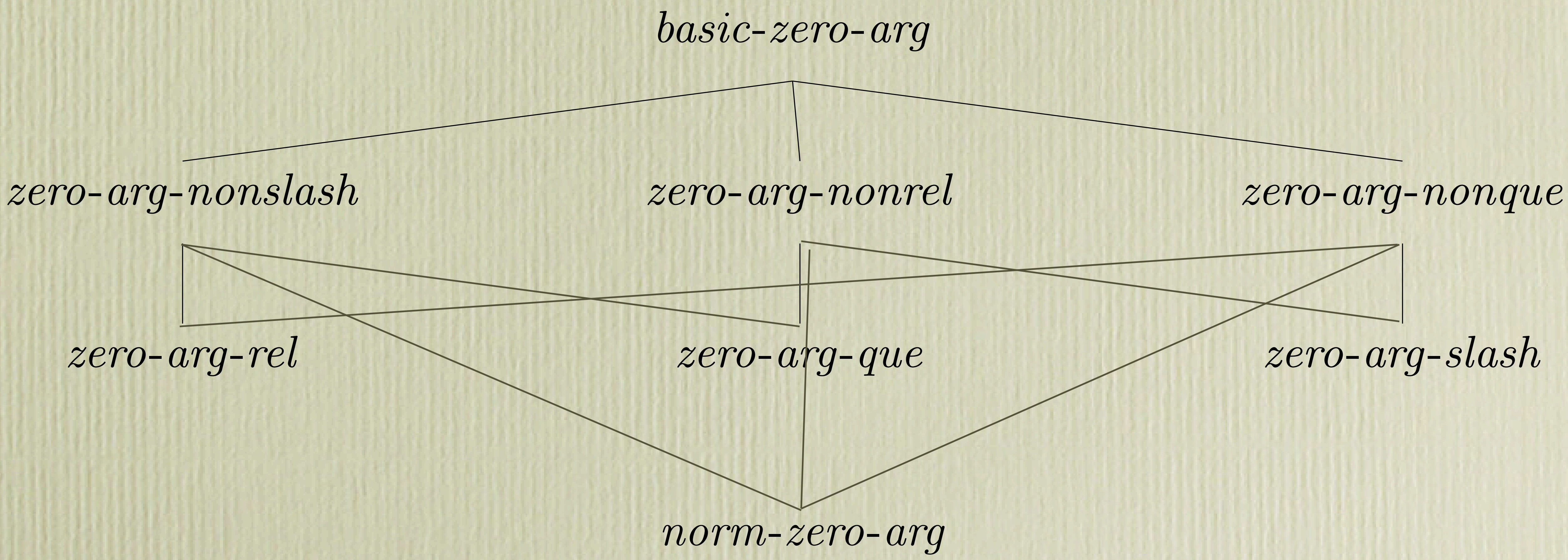
Number of arguments/ non-local features



Number of arguments/ non-local features

```
basic-two-arg := lex-item &
  [ SYNSEM [ LOCAL.ARG-S < [ NON-LOCAL [ SLASH [ LIST #smiddle,
                                                  LAST #slast ],
                                          REL [ LIST #rmiddle,
                                                LAST #rlast ],
                                          QUE [ LIST #qmiddle,
                                                LAST #qlast ] ] ],
                           [ NON-LOCAL [ SLASH [ LIST #sfirst,
                                                 LAST #smiddle ],
                                          REL [ LIST #rfirst,
                                                LAST #rmiddle ],
                                          QUE [ LIST #qfirst,
                                                LAST #qmiddle ] ]]>,
             NON-LOCAL [ SLASH [ LIST #sfirst,
                                 LAST #slast ],
                         REL [ LIST #rfirst,
                               LAST #rlast ],
                         QUE [ LIST #qfirst,
                               LAST #qlast ] ] ] ].
```

Number of arguments/ non-local features



Number of arguments/ non-local features

```
basic-zero-arg := lex-item &
  [ SYNSEM.LOCAL.ARG-S < > ].

zero-arg-nonslash := lex-item &
  [ SYNSEM.NON-LOCAL.SLASH 0-dlist ].

zero-arg-nonrel : = lex-item &
  [ SYNSEM.NON-LOCAL.REL 0-dlist ].

zero-arg-nonque : = lex-item &
  [ SYNSEM.NON-LOCAL.QUE 0-dlist ].
```

Subcategorization and linking

- What kinds of arguments does a head select?
- How are the arguments linked to semantic roles?
- Argument kinds are distinguished semantically (referential arguments v. clausal arguments).
- Semantic roles are simply numbered, and semantically linked arguments are linked in order.
- Linking is all through ARG-S, leaving valence features to particular grammars

Subjects and objects

- *intransitive*: Kim slept.
- *expletive only:* It rained.
- *transitive:* Kim eats lunch.
- *ditransitive:* Kim gave Sandy a book.

Subjects and objects

- *clausal intransitive:* That Kim sleeps is obvious.
- *clausal transitive 1:* That Kim sleeps surprises Sandy.
- *clausal transitive 2:* Sandy believes that Kim sleeps.

Subjects and objects

- *Clausal ditransitive:* Kim told Sandy that Pat slept.
- *Clasual expletive argument:* It is obvious that Kim sleeps.
- *First argument raising:* Kim seems to sleep.
- *First argument control:* Kim tries to sleep.

Subjects and objects

- *Ditrans 1st arg raising:* Kim appears to Sandy to sleep.
- *Ditrans 1st arg control:* Kim promised Sandy to leave.
- *Ditrans 2nd arg raising:* Kim believed Sandy to have left.
- *Ditrans 2nd arg control:* Kim appealed to Sandy to leave.

Subjects and objects

```
ditrans-first-arg-raising-lex-item := basic-three-arg &
[ SYNSEM [ LOCAL.ARG-S <[LOCAL.CONT.HOOK.INDEX #ind1],
                        [LOCAL.CONT.HOOK.INDEX #ind2],
                        [LOCAL.CONT.HOOK [XARG #ind1,
                                          LTOP #ltop]]>,
           LKEYS.KEYREL [ ARG1 #ind2,
                          ARG2 #ltop ]]].

ditrans-first-arg-control-lex-item := basic-three-arg &
[ SYNSEM [ LOCAL.ARG-S <[LOCAL.CONT.HOOK.INDEX #ind1],
                        [LOCAL.CONT.HOOK.INDEX #ind2],
                        [LOCAL.CONT.HOOK [XARG #ind1,
                                          LTOP #ltop]]>,
           LKEYS.KEYREL [ ARG1 #ind1,
                          ARG2 #ind2,
                          ARG3 #ltop ]]].
```

Specifiers (and subjects/ objects)

- *Specifier plus one argument:* a book about dogs, very fond of Kim
- *Specifier plus clausal argument:* the fact that Kim left, very happy to be here
- *Specifier plus raising:* Kim is completely eager to please

Parts of speech

- Encode semantic generalizations about lexemes from different parts of speech.
- Underspecify HEAD values, as the exact shape of the *head* subhierarchy seems likely to be language dependent.

Parts of speech

```
basic-verb-lex := norm-sem-lex-item &
  [ SYNSEM.LKEYS.KEYREL event-relation ].

basic-adjective-lex := norm-sem-lex-item &
  [ SYNSEM.LKEYS.KEYREL event-relation ].

basic-adposition-lex := norm-sem-lex-item &
  [ SYNSEM.LKEYS.KEYREL prep-mod-relation ].

basic-adverb-lex := norm-sem-lex-item &
  [ SYNSEM.LKEYS.KEYREL adv-relation ].

basic-noun-lex := norm-sem-lex-item &
  [ SYNSEM.LKEYS.KEYREL noun-relation ].
```

Parts of speech

```
basic-determiner-lex := norm-hook-lex-item &
  [ SYNSEM [ LOCAL [ CAT.VAL.SPEC.FIRST.LOCAL.CONT.HOOK
                                                     [INDEX #ind,
                                                      LTOP #larg],
                     CONT [ HCONS <! qeq &
                                      [ HARG #harg,
                                        LARG #larg ] !>,
                            RELS 1-dlist ] ],
              LKEYS.KEYREL quant-relation &
                           [ ARG0 #ind,
                             RSTR #harg ] ] ].
```

Parts of speech

```
basic-subord-conjunction-lex := basic-one-arg &
[SYNSEM.LOCAL[ARG-S <[LOCAL.CONT.HOOK.LTOP #ltop1]>,
              CAT.HEAD.MOD <[LOCAL.CONT.HOOK.LTOP #ltop2]>,
              CONT [ HCONS <! qeq &
                           [ HARG #harg,
                             LARG #larg ] !>,
                     RELS <! relation,
                            message &
                            [ LBL #msg,
                              PRED proposition_m_rel,
                              MARG #harg ] !>,
                     HOOK [ LTOP #msg ] ] ],
       LKEYS.KEYREL subord-relation &
                    [ LBL #larg,
                      L-HNDL #ltop1,
                      R-HNDL #ltop2 ] ] ].
```

Lexical types: Summary

- Despite the success of the Matrix so far, its usefulness has been limited by the lack of lexical resources.
- These initial hypotheses should make the initial start-up of a new grammar even faster.
- The supplied types will also serve as models for such additional types as will be needed.
- Some of these types will undoubtedly need to be refined and/or moved off to modules.

Future work: Montage

- Leverage advances in grammar engineering for documenting grammars of endangered languages
- Pair with tools for corpus annotation and descriptive grammar work
- Create accessible and persistent resources for linguistic research (Bird & Simons 2003)
- Stringent test of universals in the Matrix

Three levels of linguistic description

- Corpus annotation
- Electronic descriptive grammars
- Implemented formal grammars

Corpus annotation

- Annotated transcribed texts for grammatical information: conditional sentence, past-tense verb, etc.
- Software provides intuitive interface and creates XML (for portability)

Corpus annotation

- Linked to linguistic ontologies such as GOLD (Farrar & Langendoen 2003)
- Computer-assisted annotation: machine suggests further candidates
- Interface to lexicon software, e.g. FIELD (http://emeld.org/tools/fieldinput.cfm)

Electronic descriptive grammars

- Provide "views" on corpus examples to put relevant data at the linguist's fingertips
- Facilitate creation of web-based grammars where readers can "click through" to find all the corpus examples annotated for each phenomenon
- Facilitate output suitable for printing as a book

Electronic descriptive grammars

- Make web-published grammars discoverable to external searches through linguistic ontologies
- (Grammars only published if the author and the community so wish)
- Semi-automated testing of analyses against annotated corpus examples

Implemented formal grammars

- HPSG: Express generalizations across larger and larger sets of constructions/phrases/words
- Again indexed via a linguistic ontology
- Provide more extensive hypothesis testing, within and across languages
- Suitable for use in machine (assisted) translation, computer assisted language tutors

Implemented formal grammars

- Based on the Matrix and similar encodings of grammar engineering best practice, automate as much as possible
- First steps: Induce underspecified grammars from labeled bracketings and lexical information
- Long run: “Wizards” which customize Matrix types based on parametric questions

Summary

- The Grammar Matrix aids in the rapid start up of precision grammars
- Matrix grammars are all compatible with the same software for NLP applications
- The addition of lexical types to the Matrix should significantly increase its usefulness
- Future work: Montage will leverage the Matrix for language documentation and serve as a stringent test of Matrix hypotheses about universals