Update: The Grammar Matrix

Scott Drellishak

University of Washington

July 31, 2008

[1]This work is supported by NSF grant BCS-0644097, a gift to the Turing Center by the Utilika Foundation, and the Max Planck Institute for Evolutionary Anthropology

The Matrix

- Support building software grammars of natural languages
- HPSG syntax, MRS semantic representation, compatible with LKB
- Two parts:
  1. Universal "core" Matrix
  2. Libraries for phenomena that are widespread, but not universal

Customization System

- Users answer a web-based typological questionniare
- The customization system produces a starter grammar that parses and generates the language described
- CGI script, runs under Apache or IIS, mostly written in Python, with some Javascript and shell script
- Old sections: Language, Word Order, Sentential Negation, Coordination, Matrix Yes/No Questions, Lexicon, Test Sentences
- New! Sections: Number, Person, Gender, Other Features, Case, Direct-inverse, and a greatly enhanced Lexicon page

What's New in Customization?

- Iterators: unbounded repeating parts of the questionnaire
- Inflection: Kelly O'Hara's MA thesis work
- Case, first an initial version for Emily's class, and now a final(-ish) version
- Features: Gender, Number, Person, and Other
- Lexicon: multiple lexical types, multiple stems per type, multiple slots with multiple morphemes, and features, features, everywhere
- Lots of other things: Emily's unit test framework, validation notes on asterisks, automatic fill-in of combo boxes, archived choices files, restructured code, local variables in Javascript...

Iterators

- Formerly: exactly two nouns, two verbs, two coordination strategies...
- That's not how language works
- Now: Iterating regions, unlimited number of nouns, verbs, etc.
- Code to produce repeating regions of the questionnaire (both server-side in Python and client-side in Javascript) and to save/load the associated answers
- Allows greatly enhanced Lexicon page, which includes...

Inflection

- Kelly's code for modeling the attachment, ordering, and interactions between affixes
- Implementation:
  - Each "slot" is a lexical rule that takes another slot or a lexical type as its input, and specifies order, optionality, and interactions
  - Each "morpheme" is a subtype of its slot's type, and specifies orthography and features
- [demo]

Case

- (See my HPSG talk for more details)
- User chooses from a list of possible language types, gives the user a case hierarchy and arg structure patterns
- [demo]

Features

- Gender is done (or close to done...)
- Person and Number are the next thing I'm working on
- [demo]

Hierarchies

- New Python class for featureless type hierarchies (e.g. case, gender)
- Stores and writes out hierarchies
- Also augments hierarchies:
  - Figures out the "leaf" types
  - If needed, can create new intermediate (non-leaf) types to cover ranges of types
- Augmentation is used in direct-inverse
- Soon will be used for multiply-selectable feature values

Issues and Possible Enhancements

- Free-form questions (gender) or narrower, typologially-based questions?
  - Free-form: easier to implement, less chance of leaving something out, BUT requires more analysis from users
  - Typologically-based: easier for users to find the choice describing their language, BUT much more research required, danger of missing a possibility
- Where do questions go?
  - Convenient to have users describe features in their languages first, then use those later
  - Implied order to the questionnaire
  - Do word order questions go on the Word Order page, or on the Lexicon page?
- Split up the Lexicon page?
- Validation improvements. Real time flashing asterisks?
- Move away from fill-out-and-submit to AJAX?