Grammar files, PRED values, clause types, illocutionary force

Ling 567
February 10, 2015

Overview

- Grammar files, instances v. types
- PRED values
- Tdl style
- Illocutionary force
- Embedded clauses
- Non-verbal predicates
- Lab 6 overview
- Trigger rules

Grammar files

- matrix.tdl, head-types.tdl: Type files (core grammar)
- my_language.tdl: Type file (language specific)
- rules.tdl: Instance file for phrase structure rules
- irules.tdl: Instance file for spelling changing lexical rules
- lrules.tdl: Instance file for non-spelling changing lexical rules
- lexicon.tdl: Instance file for lexical entries
- roots.tdl: Instance file for root condition(s)
- labels.tdl: Instance file for node labels
- trigger.mtr: Instance file for trigger rules for generation
- my_langauge-pet.tdl: Grammar spec file for compilation with ‘flop’
- lkb/, ace/, pet/: Directories of files for lkb/ace/pet interaction

Roots, Labels

- Why do we sometimes see ADJ or CP as the label on the root node?

Roots, Labels

- Why do we sometimes see ADJ or CP as the label on the root node?

```
adj-label := label &
        [ SYNSEM.LOCAL[ CAT.HEAD adj,
                        COORD-STRAT "" ],
         LABEL-NAME "ADJ" ].

cp-label := label &
        [ SYNSEM.LOCAL.CAT [ HEAD comp,
                             VAL.COMPS < > ],
          LABEL-NAME "CP"].
```

Types v. instances

- Types define the feature geometry, possibilities for unification, and constraints inherited by instances.
- Instances are what the LKB actually uses to parse and generate.
- Types can have multiple supertypes.
- Instances can only inherit from one type.
- Types and instances exist in separate name spaces.

Features and types

- Features can only be “declared” for one type. Any type mentioning that feature must inherit from the declaring supertype.

- Features can only be “declared” at the outermost level.

- Good:

```
type1 := supertype &
  [ FEATURE BOOL ].

type2 := type1 &
  [ FEATURE + ].
```

- Bad:

```
type2 := supertype &
  [ FEATURE + ].

type3 := type1 &
  [ PATH.NEW-FEAT + ].
```

PRED values

- For the MT exercise, we need to coordinate on pred values.
- Convention is _English-lemma_pos_rel, where pos is drawn from {n, v, q, a, p}
- Grammar types don’t have leading underscore: exist_q_rel
- Featural information isn’t replicated in PRED values: *_went_v_rel, *_the_q_rel

Tdl style: Bad

```
demonstrative-determiner-lex := determiner-lex-supertype &
  [ SYNSEM.LOCAL.CONT.RELS
        <!
          [ PRED "exist_q_rel" ],
            #altkeyrel & arg1-ev-relation &
    [ LBL  #lbl,
      ARG1 #index ]
   !>,
  SYNSEM.LKEYS.ALTKEYREL #altkeyrel,
  SYNSEM.LOCAL.CAT.VAL.SPEC.FIRST.LOCAL.CONT.HOOK[ INDEX #index &
                                                    [ COG-ST acti+fam ]
                                                   LTOP #lbl] ].
```

Tdl style: Good

```
demonstrative-determiner-lex := determiner-lex-supertype &
   [ SYNSEM [ LOCAL [ CONT.RELS <! [ PRED "exist_q_rel" ],
                                    #altkeyrel & arg1-ev-relation &
                                    [ LBL  #lbl,
                                      ARG1 #index ] !>,
                      CAT.VAL.SPEC.FIRST.LOCAL.CONT.HOOK [ INDEX #index &
                                                           [ COG-ST activ+fam ],
                                                            LTOP #lbl ]],
                 LKEYS.ALTKEYREL #altkeyrel ]].
```

Illocutionary force: Why clausal semantics?

- Illocutionary force correlates with syntactic form.
- MRS representations should include all semantic information that is syntactically marked.

Aside: Perlocutionary, Locutionary, Illocutionary

- Locutionary act: The act of saying something
- Illocutionary act: The act of asking, asserting, commanding, etc. by saying something
- Perlocutionary act: The act of getting someone to do or believe something by asking, asserting, etc. something.

What’s a clause?

- Syntactically complete
- Expresses some illocutionary force
- Contrasts with fragments, some of which can also carry illocutionary force
- Marking of illocutionary force is often associated with either the clause as a whole or with its head verb
- Clauses can be matrix or embedded
- Embedded clauses can be modifiers or arguments
- Embedded clauses can carry illocutionary force, too

Our general strategy

- Represent illocutionary force with a feature of events called ‘SF’.
- Possible values of SF: command, prop-or-ques, proposition, question
- For Matrix clauses, non-branching rules at the top of the tree set SF depending on syntactic features.
  - OR: Subject attaching rules constrain SF.
  - OR: Other characteristic rules/lex items constrain SF.
- For embedded clauses, elements higher up the tree (complementizers, selecting verbs) or unary constructions constrain SF.

Marking of embedded clauses

- Just like matrix clauses
- Special verbal inflection
- Complementizers
- Different word order
- ... others?

- The feature [MC bool] can be helpful here

Non-verbal predicates

- This section deals with sentences that have a “copula” verb in some languages and no verb at all in others.
- APs/PPs have a semantic role available
  - Required copula: Treat it as a raising verb
  - No copula: Let the APs/PPs be heads in the head-subj rule
- NPs are semantically saturated
  - Required copula: Different lex entry that introduces _be_v_id_rel
  - No copula: Non-branching rule that introduces _be_v_id_rel and the subject requirement

Non-verbal predicates

- Some languages have a copula variably:
  - Across all contexts
  - Only with NPs, but not APs/PPs (etc)
  - Only in certain tenses
- First two can be handled with just appropriate combinations of the strategies discussed
- To get restriction to certain tenses, need to add constraints to the copula and/or the lexical or phrase structure rules involved in licensing verbless clauses.

Non-verbal predicates

- Locative NPs
  - Some languages use NPs inflected with a particular case where others use PPs (as both modifiers and predicates)
  - We'll only worry about the predicative use (for now)
  - The strategy we'll take involves a non-headed unary rule that builds a PP out of a [CASE loc] NP.
    - Why non-headed?
    - Why not do this with a lexical rule?

Lab 6

- Check that matrix polar questions are working, and debug as necessary
- Add sentential complement verbs
- Get sentences with NP, PP, and AP predicates working
- Make sure MRSs are correct, and debug as necessary
- Make sure your grammar can *generate* (as well as parse), and debug as necessary

Lab 6 reminders

- Your write up should illustrate each analysis with IGT examples *that parse with the grammar you turn in.*

- You should test your grammar both with individual sentences one at a time in the LKB and with [incr tsdb()] processing of the whole test suite.

- Use [incr tsdb()] to see which examples are ambiguous according to the grammar, and check to see if the ambiguity is justified.

- Incremental development: If you have lots of similar items to enter, get one working first, then enter the rest.

Overview

- Grammar files, instances v. types
- PRED values
- Tdl style
- Illocutionary force
- Embedded clauses
- Non-verbal predicates
- Lab 6 overview
- Trigger rules

Trigger rules

- Semantically empty lexical entries cause headaches on generation
- Let them all in as often as the parser wants them: exploded search space
- Keep them all out: somethings won’t parse
- Solution: trigger rules (trigger.mtr)
- http://moin.delph-in.net/LkbGeneration