May 17, 2004

Clause types
Overview

- Distinguish syntactic form, semantic content, pragmatic use
- Message types in the Matrix
- Messages in MRS
- Parameters of wh questions
- HOOK.MSG
- Preview: Long distance dependencies
A three-way distinction (1/2)

1. Syntactic clause types
2. Semantic content (message) types
3. Speech act types
   - Clauses of different types express different kinds of semantic messages, which can be used in different speech acts.
A three-way distinction (2/2)

<table>
<thead>
<tr>
<th>Clause type</th>
<th>Message type</th>
<th>Speech act</th>
</tr>
</thead>
<tbody>
<tr>
<td>declarative</td>
<td>proposition</td>
<td>assertion</td>
</tr>
<tr>
<td>interrogative</td>
<td>question</td>
<td>query</td>
</tr>
<tr>
<td>imperative</td>
<td>outcome</td>
<td>command</td>
</tr>
<tr>
<td>exclamative</td>
<td>fact</td>
<td>exclamation</td>
</tr>
<tr>
<td>declarative</td>
<td>fact</td>
<td>–</td>
</tr>
<tr>
<td>interrogative</td>
<td>coerced fact reading</td>
<td>–</td>
</tr>
</tbody>
</table>
Why have a three way distinction?

- Theoretical clarity: syntactic structure, semantic content and illocutionary force are three different things.

- Ability to explore mismatches:
  - Many different clause types can express facts
  - Embedded clauses don’t have illocutionary force
Why should this be implemented?

- Linguistic hypothesis testing (constructions pair form and meaning, what kinds of forms, what kinds of meanings, how do they map)

- Practical applications — Examples?
Messages in the Matrix (1/2)

- Implemented as another type of relation, with a PRED value distinguishing message types, a LBL, and a single handle-valued argument which points to the state of affairs.

  ```
  basic_message := relation.
  message := basic_message &
              [ PRED message_m_rel,
                MARG handle ].
  no-msg := basic_message.
  ```

- The fourth HOOK feature, MSG, points to the message, if any, of a sign, for selectional restrictions.
Messages in the Matrix (2/2)

predsort
  |
message_m_rel
    |
command_m_rel  prop-or-ques_m_rel
      |
proposition_m_rel  abst-ques_m_rel
        |
question_m_rel  ne_m_rel
Example MRS with message

< h1, e2,
{ h1:proposition_m_rel(h5),
  h6:def_q_rel(x9,h8,h7),
  h10: _dog_n_rel(x9),
  h11:_bark_v_rel(e2,x9)},
{ h5 qeq h11, h8 qeq h10 } >
Questions always embed propositions (1/2)

- Provides a tenable formalization of the idea that questions are open propositions (Ginzburg & Sag 2000:108–109).

- Thus an MRS with a question_m_rel will always also have an proposition_m_rel as the argument of the question_m_rel.
Questions always embed propositions (2/2)

< h1, e2,
{ h1:question_m_rel(h5),
  h5:proposition_m_rel(h6),
  h9:def_q_rel(x12,h11,h10),
  h13:_dog_n_rel(x12),
  h14:_bark_v_rel(e2,x12)},
{ h6 qeq h14, h11 qeq h13 } >
Messages on embedded clauses/no scope ambiguity

< h1, e2,
{ h1:proposition_m_rel(h5),
  h6:pronoun_n_rel(x7:1sg), h8:pronoun_q_rel(x7,h9,h10)
  h11:_know_v1_rel(e2,x7,h12), h12:proposition_m_rel(h15)
  h16:def_q_rel(x19,h18,h17), h20: _dog_n_rel(x19),
  h21: _bark_v_rel(e22,x19)},
{ h5 qeq h11, h9 qeq h6
  h15 qeq h21, h18 qeq h20 } >
Parameters and wh thingies (1/2)

- Wh expressions are taken as marking the parameters of a wh question.
- Ginzburg and Sag argue that they are not quantifiers.
Parameters and wh thingies (1/2)

- If there are multiple questions a sentence, parameters have some freedom (subject to syntactic constraints) as to which question the go with:
  - Who wonders who saw what?
  - Who wondered about the answer to the question who saw what?
  - For which persons x and objects y, did x wonder who saw y?
  - *For which persons x and y, did x wonder what y saw?*
  - Who wondered what was seen by WHO?
Using *HOOK.MSG*

- The HOOK feature MSG records the message of the sign.

- If there is no message (i.e., the sign is non-clausal), the MSG value should be *no_msg*.

- Selecting predicates can check the MSG value, e.g.:
  
  - I know whether Kim left/that Kim left.
  - I believe that Kim left/*whether Kim left.
  - I wonder whether Kim left/*that Kim left.
Supplying MSG values

- Two basic strategies:
  - Cross-classify clause types with phrase types.
  - Provide non-branching rules which add a message value.
- Combinations of these strategies are possible.
- Convenience of each is going to depend on the language:
  - If subjects are realized after objects, it’s convenient to have the head-subj rule be a type of clause.
  - If subjects can be realized before objects, less so.
Syntax of clause types crosslinguistically

- So far we’ve been dealing with propositions only, and primarily matrix clauses only.
- How does your language express matrix yes-no questions?
- How does your language express embedded yes-no questions?
- How does your language express embedded propositions?
- Find at least one verb that can embed a (finite) interrogative clause.
- Find at least one verb that can embed a finite declarative clause.
Many languages allow dislocation of certain constituents to the (left) edge of a clause.

In many languages, such dislocation is an option in the expression of wh-questions. Examples?

HPSG doesn’t treat these via movement, but rather by a feature SLASH.
Distinguish bottom, middle and top of a LDD.

- Bottom: Something’s missing, record that fact.
- Middle: Something’s missing, pass up that information.
- Top: Head-filler construction pairs a filler with a constituent with a matching gap.