MRS

Ling 567
February 3, 2025

Overview

- MRS
  - Goals, design principles
  - Flat semantics
  - Underspecified quantifier scope
  - Linguistic questions
  - MRS in feature structures
- Lab 5 preview

We’ve arrived at MRS!

- Flat structure
- Underspecification & partial specification of quantifier scope are possible

Linguistic Questions

- How do we build MRS representations compositionally?
- Is it linguistically adequate to insist that no process suppress relations?
- Under what circumstances do NLs (partially) constrain scope?
- Is it linguistically adequate to give scopal elements (esp. quantifiers, but also scopal modifiers) center-stage?

MRS in feature structures

- RELS: List (append-list) of relations
- HCONS: List (append-list) of handle constraints
- ICONS: List (append-list) of individual constraints
- HOOK: Collection of features ‘published’ for further composition: INDEX, LTOP, XARG
- ARGn: Roles within relations

Quick comparison to 566

- SWB RESTR = Matrix RELS
- SWB INDEX = Matrix HOOK.INDEX
- New here:
  - HCONS, ICONS
  - HOOK.LTOP, HOOK.XARG
  - C-CONT

Anatomy of an MRS

- An MRS consists of:
    - A top handle
    - A list of relations, each labeled by a handle
    - A list of handle constraints
    - (A list of individual constraints)
    - An (underspecified) MRS is well-formed iff the constraints can be resolved to form one or more trees (singly-rooted, connected, directed acyclic graphs).

Anatomy of a relation

- A relation has:
  - A predicate (string or type)
  - A label (handle)
  - One or more arguments: ARG0-n (ARG0 canonically being the event or individual introduced by the relation)

- The value of each ARGn is either:
  - An index, canonically identified with the ARG0 of another relation
  - A handle: identified with the label of another relation, the HARG of a handle constraint, or not identified with anything

Anatomy of a handle constraint

- Current sole handle constraint type: qeq
- ‘Equal modulo quantifiers’
- Features: HARG, LARG
- → Unless some quantifier scopes in between, the value of this ARGn is the same as the label of that relation.
- When the label of a relation is the value of an ARGn, this corresponds to a branch in an MRS tree.
- When the value of an ARGn is qeq the label of a relation, this corresponds to a ‘dotted’ branch – i.e., a dominance relation.

When else are handles identified?

- Relations with the same handle value share the same scope.
- Typically, we see this with non-scopal modifiers (adverbs, adjectives, PPs) which share their handles with their modifiees.

Composition: Overview

- RELS and HCONS (and ICONS) on mother nodes
- HOOK, LKEYS
- ARGn <> indices
- ARGn <> handles
- LBL <> LBL
- Building *qeq*s

RELS and HCONS on mother nodes

- The RELS and HCONS (and ICONS) value of the mother is the append of the values from the daughter(s) and the C-CONT of the mother.

- C-CONT is the ‘constructional content’: allows phrase structure rules to introduce relations.

- Examples?

- From a semantic point of view, the C-CONT is just another daughter.

Semantic compositionality in action

```
basic-unary-phrase := phrase &
  [ STEM #stem,
    SYNSEM [ L-PERIPH #lperiph,
             R-PERIPH #rperiph,
             LOCAL [ CAT.MKG #mkg,
                     CONT [ RELS.APPEND < #r1, #r2 >,
                            HCONS.APPEND < #h1, #h2 >,
                            ICONS.APPEND < #i1, #i2 > ] ] ],
    C-CONT [ RELS #r1,
             HCONS #h1,
             ICONS #i1 ],
    ARGS < sign & [ STEM #stem,
             SYNSEM [ L-PERIPH #lperiph,
                      R-PERIPH #rperiph,
                      LOCAL local &
                            [ CAT.MKG #mkg,
                              CONT [ RELS #r2,
                                     HCONS #h2,
                                     ICONS #i2 ] ] ] ] > ].
```

Now what?

- Phrase structure rules (and lexical rules) gather up RELS and HCONS from daughters.

- Phrase structure rules also (optionally) introduce further RELS and HCONS.

- How do we link the ARGn positions of the relations to the right things?

- How do we link the HARG/LARG of qeqs to the right things?

HOOK

- The CONT.HOOK is the information that a given sign exposes for further composition.
- By hypothesis, this includes only:
  - INDEX (the individual or event denoted by the sign, linked to some ARG0)
  - LTOP (the local top handle of the sign)
  - XARG (the external argument of the sign)
- The HOOK of a sign is identified its with the C-CONT.HOOK.
- The C-CONT.HOOK in turn is identified with the semantic head daughter, if there is one.
  - Otherwise, the LTOP, INDEX, and XARG inside C-CONT.HOOK need to be constrained appropriately.

LKEYS

- The feature LKEYS houses pointers to important relations on the RELS list, most notably LKEYS.KEYREL.

- Only appropriate for lexical items.

- Serves as a uniform place to state linking constraints.

- Linking constraints: equality between HOOK.INDEX or HOOK.LTOP of arguments/modifiees and LKEYS.KEYREL.ARGn.

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

ARGn <> indices

```
intransitive-lex-item := basic-one-arg-no-hcons &
   [ ARG-ST < [ LOCAL.CONT.HOOK.INDEX ref-ind &
                                        #ind ] >,
     SYNSEM.LKEYS.KEYREL.ARG1 #ind ].

intersective-mod-lex := no-hcons-lex-item &
   [ SYNSEM [ LOCAL.CAT.HEAD.MOD
                        < [ ..INDEX #ind ]] >,
              LKEYS.KEYREL.ARG1 #ind ] ].
```

ARGn <> handles (1/2)

```
clausal-second-arg-trans-lex-item := basic-two-arg &
[ ARG-ST < [ LOCAL.CONT.HOOK.INDEX ref-ind & #ind ],
           [ LOCAL.CONT.HOOK.LTOP #larg ] >,
  SYNSEM [ LOCAL.CONT.HCONS <! qeq &
                               [ HARG #harg,
                                 LARG #larg ] !>,
           LKEYS.KEYREL [ ARG1 #ind,
                          ARG2 #harg ] ] ].
```

ARGn <> handles (2/2)

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

LBL <> LBL

```
isect-mod-phrase :=
  head-mod-phrase-simple &
  head-compositional &
   [ HEAD-DTR.SYNSEM.LOCAL.CONT.HOOK.LTOP #hand ],
     NON-HEAD-DTR.SYNSEM.LOCAL.CONT.HOOK.LTOP #hand
```

- The rule for non-scopal modifiers identifies the LTOP of the two daughters, and thus the LBL of the main relation introduced by each.

- The HOOK value of the whole thing comes from the syntactic head, thanks to the type head-compositional.

Scopal modifiers (1/2)

```
scopal-mod-phrase :=
    head-mod-phrase-simple &
  [ NON-HEAD-DTR.SYNSEM.LOCAL
      [ CAT.HEAD.MOD < [ LOCAL scopal-mod ] >,
        CONT.HOOK #hook ],
    C-CONT [ HOOK #hook,
             HCONS <! !> ] ].
```

- No identification of LTOPs.
- Non-head (adjunct) daughter is the semantic head.

Scopal modifiers (2/2)

```
scopal-mod-lex := lex-item &
  [ SYNSEM [ LOCAL [
      CAT.HEAD.MOD < [ LOCAL scopal-mod &
                         [ ..LTOP #larg ]] >,
    CONT.HCONS <! qeq &
                  [ HARG #harg,
                    LARG #larg ] !> ],
    LKEYS.KEYREL.ARG1 #harg ]].
```

- Builds qeq between its ARG1 and the MOD's LTOP

Building qeqs

- Determiners
- Scopal adverbs
- Clausal complement verbs (and nouns, adjectives, adpositions...)

Summary

- Phrase structure and lexical rules:
  - ... gather up RELS and HCONS (and ICONS)
  - ... potentially add further RELS and HCONS
  - ... unify elements on valence/mod lists with signs
  - ... pass up and/or modify HOOK information
- Lexical entries:
  - ... orchestrate the linking between valence/mod lists and the ARGn positions in the relations they contribute
  - ... expose certain information in the HOOK

Composition: Overview

- RELS and HCONS (and ICONS) on mother nodes
- HOOK, LKEYS
- ARGn <> indices
- ARGn <> handles
- LBL <> LBL
- Building *qeq*s

Overview

- MRS
  - Goals, design principles
  - Flat semantics
  - Underspecified quantifier scope
  - Linguistic questions
  - MRS in feature structures
- Lab 5 preview