# Case, Agreement, Argument Optionality, Tense/ Aspect/Mood

Ling 567
April 16, 2019

## Lab 3 preview

- http://courses.washington.edu/ling567/lab3.html

## Overview

- Case
- Agreement
- Pronouns
- Argument optionality
- Tense/aspect/mood
- Morphotactics (reprise)
- Testsuite construction

## Case

- Changes in the form of ‘nouny’ (NP or PP) arguments based on grammatical function
- Expressed as:
  - choice of adposition
  - inflection on noun (suffix, prefix)
  - choice of pronoun form
  - choice of determiner/adjective form

## Case

- Basic HPSG analysis:
    - verbs constrain CASE of their arguments
    - nouns (or dets or Ps or…) constrain their own CASE
    - either or both can be underspecified (or partially specified)

## Case library (Drellishak 2009)

- Case marking system for core arguments
  - None, nom-acc, erg-abs, tripartite, focus case, various kinds of splits…
- Default ARG-ST frames for verbs based on case marking system
- Ability to define additional cases
- Case morphology, case-marking adps, case on determiners

## Case example (small jpn grammar)

```
nom-acc-transitive-verb-lex := transitive-verb-lex &
  [ ARG-ST < [ LOCAL.CAT.HEAD +np &
                              [ CASE nom ] ],
             [ LOCAL.CAT.HEAD +np &
                              [ CASE acc ] ] > ].


acc-marker := case-marking-adp-lex &
  [ STEM < "を" >,
    SYNSEM.LOCAL [ CONT [ HOOK [ ICONS-KEY.IARG1 #clause,
                                 CLAUSE-KEY #clause ],
                          ICONS <!  !> ],
                   CAT.HEAD.CASE acc ] ].
```

## Argument optionality

- There are many cases in which an argument may be semantically present but syntactically absent
- Semantically, these cases can be categorized by how the missing argument is interpreted
- Syntactically, they can be categorized by how the missing argument is licensed

## Overview

- Case
- Agreement
- Pronouns
- Argument optionality
- Tense/aspect/mood
- Morphotactics (reprise)
- Testsuite construction

## Agreement (Drellishak 2009)

- PNG features as features of semantic indices
- Values are language-specific

```
matrix.tdl

ref-ind := index & event-or-ref-index &
   [ PNG png ].

png := avm.

japanese.tdl

png :+ [ PER person,
    NUM number ].
```

## Agreement

- Nouns/noun morphology constrains their own PNG values
- Agreeing elements constrain PNG on INDEX of SUBJ, COMPS, MOD or SPEC
- Ex from mcb grammar:

```
verb-pc1_lrt3-lex-rule := add-only-no-ccont-rule & verb-pc1-lex-
rule-super &
 [ SYNSEM.LOCAL.CAT.VAL.SUBJ.FIRST.LOCAL.CONT.
         HOOK.INDEX.PNG.GEND inanim ].
```

## Clusivity (Drellishak 2009)

- For languages with a inclusive/exclusive distinction in non-sg number
- Joined PERNUM feature rather than separate PER &
- Example from Umatilla Sahaptin grammar:

```
pernum := *top*.
sg := pernum.
1sg := 1st & sg.
2sg := 2nd & sg.
3sg := 3rd & sg.
1du := du & 1pl_excl+1du_excl+1du+1pl_incl+1du_incl+1pl.
1du_incl := 1du & 1pl_incl+1du_incl.
1du_excl := 1du & 1pl_excl+1du_excl.
2du := du & 2du+2pl.
3du := du & 3du+3pl.
1pl := pl & 1pl_excl+1du_excl+1du+1pl_incl+1du_incl+1pl.
1pl_incl := 1pl & 1pl_incl+1du_incl.
1pl_excl := 1pl & 1pl_excl+1du_excl.
2pl := pl & 2du+2pl.
3pl := pl & 3du+3pl.
non-1st := pernum.
3rd := non-1st.
2nd := non-1st & non-3rd.
2du+2pl := 2nd.
1pl_excl+1du_excl+1du+1pl_incl+1du_incl+1pl := 1st.
1pl_excl+1du_excl := 1pl_excl+1du_excl+1du+1pl_incl+1du_incl+1pl.
1pl_incl+1du_incl := 1pl_excl+1du_excl+1du+1pl_incl+1du_incl+1pl.
1st := non-3rd.
3du+3pl := 3rd & non-3sg.
non-3sg := pernum.
pl := non-3sg.
du := non-3sg.
non-3rd := non-3sg.
```

## Overview

- Case
- Agreement
- Pronouns
- Argument optionality
- Tense/aspect/mood
- Morphotactics (reprise)
- Testsuite construction

# Pronouns

- Nouns that don’t take determiners
- Have interesting person & number values
- Should all have the PRED value `pron_rel`
- NB: Always independent words
    - Pronoun “clitics” are probably either affixes or 2nd position auxiliaries
    - “Incorporated pronouns” are affixes (+ argument optionality)

## Overview

- Case
- Agreement
- Pronouns
- Argument optionality
- Tense/aspect/mood
- Morphotactics (reprise)
- Testsuite construction

## Semantic classification

- Indefinite null instantiation: *I ate*
  - The referent of the missing argument is indefinite, not (necessarily) recoverable from context
- Definite null instantiation: *I told you already*
  - The referent of the missing argument is definite, i.e., it is presumed to be recoverable from context
- Constructional null instantiation: *Eat!, I told Kim to eat., Stir until completely mixed.*
  - The referent of the missing argument is determined by the syntactic construction.
- Impersonal null instantiation:
  - The referent of the missing argument is “people in general”, or “one”

## Syntactic classification

- Lexically licensed: The potential for an argument to be missing is determined by the lexical type of the selecting head
  - *eat* allows indefinite null instantiation of its object
  - *devour* does not
- Systematic: The potential for an argument to be missing is determined by its grammatical function, plus (possibly) inflection on the verb as well as other factors (PNG features on the argument, TAM features on the verb, ...)

## Analysis in the Matrix

- Constructional null instantiation is handled by the analyses of imperatives, raising, etc.
- Distinction between definite and indefinite null instantiation handled by the COG-ST feature
    - Pronouns and arguments subject to DNI [COG-ST in-foc, SPECI +]
    - Arguments subject to INI are [COG-ST type-id, SPECI -]
- Posit opt-comp and opt-subj rules parallel to the bare NP rule
- Use a feature [OPT bool] to constrain possibility of argument drop/overt realization
- Use a feature [OPT-CS cog-st] to allow lexical items (and lexical rules!) to indicate COG-ST value of an argument in the event it is dropped

## The features OPT and OPT-CS

- OPT and OPT-CS are both features of *synsem*
- However, nothing constrains its own OPT or OPT-CS value
- Rather, heads may constrain certain arguments to be [OPT -] (can’t be dropped) or [OPT +] (must be dropped)
- OPT-CS is a “junk slot” to allow a lexical item to store information about how an argument will be interpreted if it’s unexpressed
- The opt-comp rule will identify the OPT-CS value and HOOK.INDEX.COG-ST values of any argument it caches out as unrealized
- Because the HOOK.INDEX of every argument is identified with an ARGn position of the head’s KEYREL, this information ends up in the MRS
- Note that we’re not associating pronouns or quantifier relations with these argument positions

## The Matrix opt-comp type

```
basic-head-opt-comp-phrase := head-valence-phrase & head-only &
                              head-compositional &
  [ INFLECTED #infl,
    SYNSEM canonical-synsem &
    [ ..CAT [ VAL [ SUBJ #subj, COMPS #comps, SPR #spr, SPEC
#spec ],
              MC #mc, POSTHEAD #ph ],
      MODIFIED #mod ],
    HEAD-DTR [ INFLECTED #infl & +,
               ..CAT [ VAL [ SUBJ #subj, SPR #spr, SPEC #spec,
                             COMPS < unexpressed &
                                   [ OPT +, OPT-CS #def,
                                     ..INDEX.COG-ST #def ] .
#comps >],
                                     MC #mc, POSTHEAD #ph ],
               ..CONT.HOOK.INDEX event,
              MODIFIED #mod ],
    C-CONT [ RELS <! !>, HCONS <! !> ] ].
```

## Argument optionality: Your tasks (if applicable)

- Check that the analysis provided by the customization system covers the syntactic facts of your language
  - If it doesn’t, post to Canvas
- Determine (as best you can) the semantic facts: How are arguments interpreted when they are dropped?
  - If object markers are usually required for object drop, do you get INI without object markers and DNI with?
- [[Constrain the OPT-CS value of droppable arguments to capture the semantic facts you discover.  This may require making multiple subtypes of verbs.]]

## Overview

- Case
- Agreement
- Pronouns
- Argument optionality
- Tense/aspect/mood
- Morphotactics (reprise)
- Testsuite construction

## Basic concepts

- Position class: A supertype to lexical rules which fit in the same slot
- Lexical rule type: *lex-rule* and its subtypes, all have DTR feature
- Lexical rule instance: A grammar entity (manipulatable by the LKB) which inherits from a lexical rule type and specifies a spelling change (including no change).
- Forbids constraint: A specification in the customization system stating that a stem lexical rule type (including a position class) cannot co-occur with another lexical rule type, instance, pc or stem.
- Requires constraint: A specification in the customization system stating that a stem lexical rule type (including a position class) must co-occur with another lexical rule type, instance, pc or stem.

## Position classes, inputs and lexical rule hierarchies

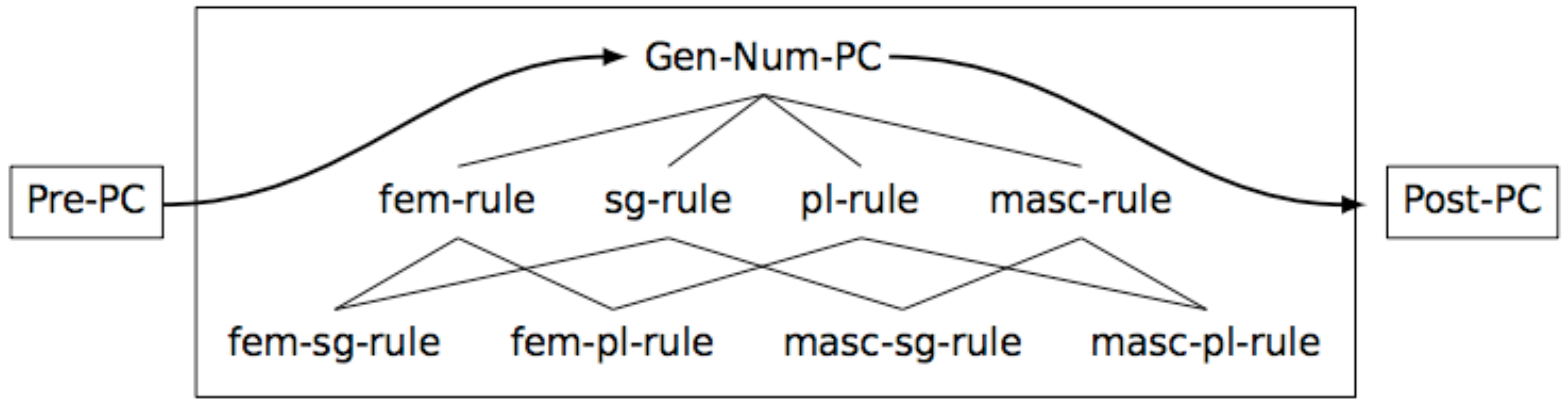

**Figure 9:** Example lexical rule type hierarchy in a position class

(Goodman 2013)

## To define a position class

- Required:
  - Whether or not it is obligatory
  - Possible inputs and prefix/suffix
    - = position in the string
- Optional:
  - Requires/forbids constraints

## To define a lex rule type

- Required
  - Nothing (though defaults fill in)
- Optional
  - Name
  - Supertype (if it doesn’t inherit directly from its position class)
  - Feature/value pairs (optional, but this is usually the point!)
  - Requires/forbids constraints

## To define a lex rule instance

- Required
    - Affix v. no affix
    - Spelling for affix
- Optional
    - Nothing

## tdl files

- matrix.tdl: Supertypes for lex-rules, which handle the copying up of everything you’re not changing
- my_language.tdl: Position classes and lex rule types defined through the customization system; features for inside INFLECTED
- lrules.tdl: Instances for non-spelling-changing lex rules (zero morphemes)
- irules.tdl: Instances for spelling-changing lex rules

## Handling of morphotactics

- Rule order handled through super types and typing the DTR feature
- Requires/forbids through the INFLECTED feature

```
case-lex-rule-super := representative-rule-dtr &
                       add-only-no-ccont-rule &
                       noun-telic-rule-dtr &
[ INFLECTED [ CASE-FLAG +,
              INNER-NEGATION-FLAG #inner-negation,
              NUMBERED-FLAG #numbered ],
  DTR case-rule-dtr &
       [ INFLECTED [ INNER-NEGATION-FLAG #inner-negation,
                     NUMBERED-FLAG #numbered ] ] ].
```

## Overview

- Case
- Agreement
- Pronouns
- Argument optionality
- Tense/aspect/mood
- Morphotactics (reprise)
- Testsuite construction

## Testsuites

- Select/construct IGT to exemplify specific properties
- Sentences should otherwise be as simple as possible
- Positive and negative examples
    - => You will likely need to construct examples on the basis of descriptions in your materials and source examples

## Testsuites

- Format specified to answer needs of both human readers and make_item.py
- make_item.py will check format & produce [incr tsdb()] file
    - It’s finicky; try early & often
    - Ask questions early & often
- http://courses.washington.edu/ling567/testsuites.html#formatting
- http://courses.washington.edu/ling567/testsuites.html#preliminaries