Incremental Deterministic HPSG parsing

Gisle Ytrestøl
University of Oslo, Department of Informatics
gisley@ifi.uio.no

July 20, 2009

Goals

- Build an incremental deterministic transition based parser for the HPSG framework
- Domain: NLP related Wikipedia articles (English)
- New HPSG Treebank: WeScience

Fundamental

- takes a sentence as input string
- outputs an HPSG representation of the sentence
- deterministic incremental (i.e. no backtracking once committed to a parse transition)
- linear-time

Incremental Deterministic

- Incremental:
  1. Input buffer β never grows in size
  2. Passive edges in the stack are never altered or removed
  3. Buffer β must be empty upon termination
- Deterministic
  1. Derives only one analysis – Maximally efficient
  2. Once committed to a parse transition, never backtrack
- May also possibly experiment with near-deterministic parsing that allows backtracking and beam search.

English Resource Grammar

- English Resource Grammar (ERG) is a broad-coverage, linguistically precise HPSG-based grammar of English (`http://www.delph-in.net/erg/`)
- Each leaf lexical type (word) is assigned a lexical entry (LE) in ERG which determines most of its properties

LE type	Annotations	Example
v_-_it_le	It-subj	It rains.

Table: Example of LE type

ERG Lexical and Syntactic Rules

> *The ERG grammar contains several types of rules, including inflectional and derivational lexical rules, affixation rules for punctuation, and syntactic constructions*

(`http://wiki.delph-in.net/moin/ErgRules`)

- The tree of ERG rules describes the HPSG representation of a sentence
- A successful parse with the incremental HPSG parser will yield the tree of ERG rules

ERG Lexical and Syntactic Rules

Daughter sequence	Annotations	Rule type

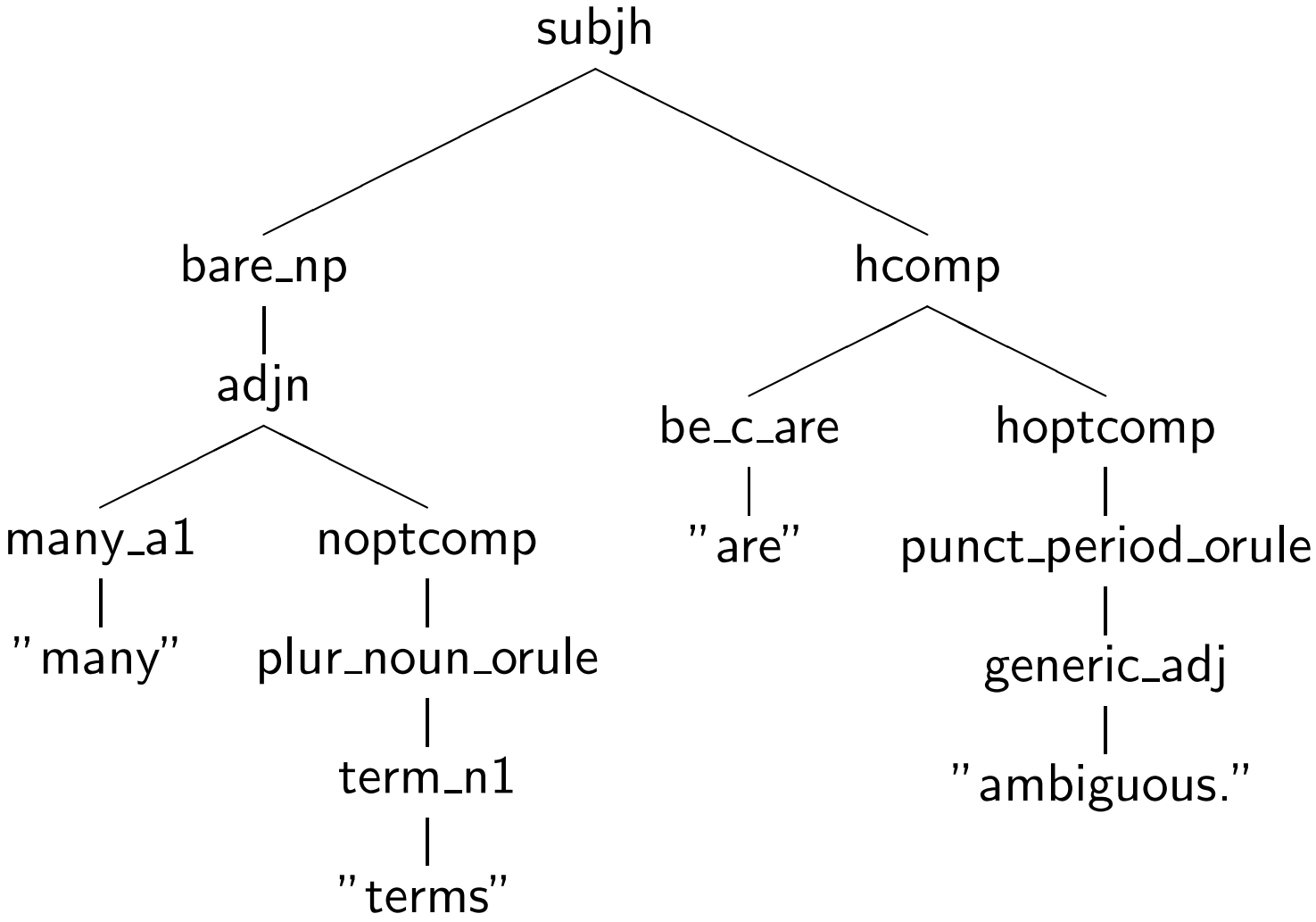
Table: Rule name template (field separated by underscore)

Rule name	Annotations	Example	Old rule name
n_pl_olr	Plural noun with |-s| suffix	cats	plur_noun_orule

Table: Example of ERG rule

Parser Layout

- Shift-reduce parsing (bottom-up)
- Input buffer of (ERG) Lexical Entries corresponding to input sentence
- Example input sentence: *Many terms are ambiguous*
- Input buffer = (many_a1, term_n1, be_c_are, generic_adj)


subjh
bare_np
hcomp
adjn
be_c_are
hoptcomp
many_a1
noptcomp
"are"
punct_period_orule
"many"
plur_noun_orule
generic_adj
term_n1
"ambiguous."
"terms"

Data Structure

Given a set R $= \{r_0, r_1, ...r_m\}$ of ERG Lexical and Syntactic rules and a sentence x $= (w_1, w_2, ...w_n)$, a parser configuration for x is a triple c $= (\alpha, \beta, \pi)$ where

- α is a stack of active edges
- β is a sorted sequence of lexical entries corresponding to sentence x
- π is a stack of passive edges
- The stack of passive edges π makes up the full HPSG representation of the input string if the string is accepted

Transition System

Every parse action can be seen as a classification task

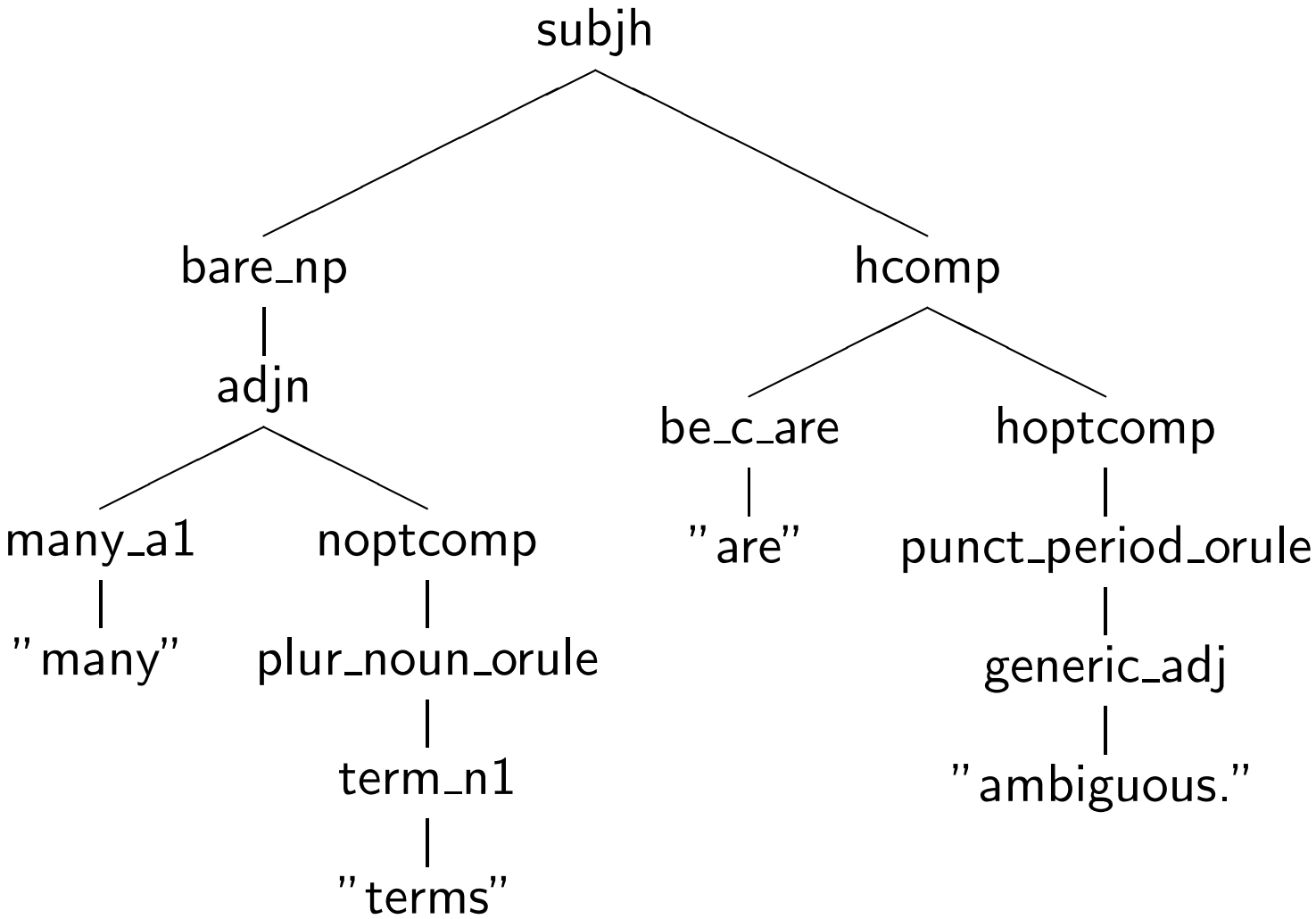
- READ (moves next ERG lexical entry from β to π)
- UNIT(C^1) (add unary mother to π(i))
- ACTIVE (add active edge to stack α)
- PASSIVE(C^2) (Build C^2 with active edge to make passive edge, and add to π)
- (ACCEPT)
  # of classification choices: # ERG rules x 2 + 2 The preconditions will reduce this number significantly.

Example

TRANSITIONS

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

```
(subjh 0 4
  (bare_np 0 2
    (adjn 0 2
      (many_a1 0 1
        ("many" ) )
      (noptcomp 1 2
        (plur_noun_orule 1 2
          (term_n1 1 2
            ("terms" ) ) ) ) ) )
  (hcomp 2 4
    (be_c_are 2 3
      ("are" ) )
    (hoptcomp 3 4
      (punct_period_orule 3 4
        (generic_adj 3 4
          ("ambiguous." ) ) ) ) ) )
10021300 many terms are ambiguous.
```


subjh
bare_np
hcomp
adjn
be_c_are
hoptcomp
many_a1
noptcomp
"are"
punct_period_orule
"many"
plur_noun_orule
generic_adj
term_n1
"ambiguous."
"terms"

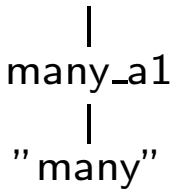
INITIAL

Triple = α (Active Edges), β (Buffer), π (Passive edges)
α: []
β: [('many_a1','many', ['0','1']),('term_n1', ""terms"', ['1', '2']), ('be_c_are', ""are"', ['2', '3']), ('generic_adj', ""ambiguous."', ['3', '4'])]
π: []
$\alpha - \beta -$ *format*: ([identifier, start-edge, end-edge, category, daugther(s) identifier])

TRANSITION: 1) READ

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

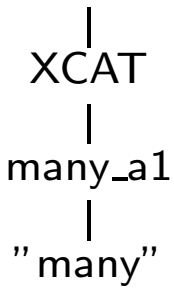
α: []
β: [('term_n1', '"terms"', ['1', '2']), ('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]])



TRANSITION: 3) ACTIVE

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

α: ([101,0,1,XCAT,[100]])
β: [('term_n1', '"terms"', ['1', '2']), ('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]])



TRANSITION: 1) READ

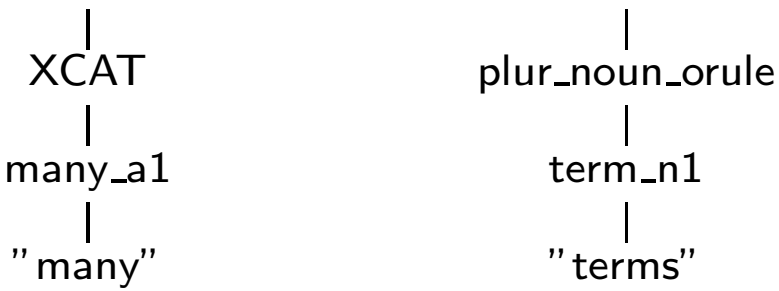
- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

α: ([101,0,1,XCAT,[100]])
β: [('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]])

TRANSITION: 2) UNIT (C^1)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

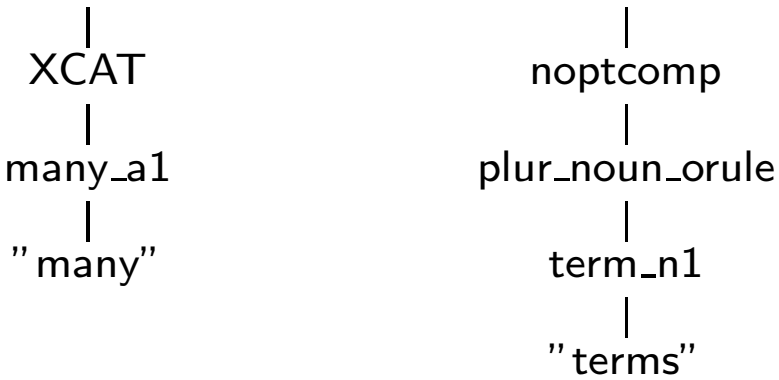
α: ([101,0,1,XCAT,[100]])
β: [('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]])



TRANSITION: 2) UNIT (C^1)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

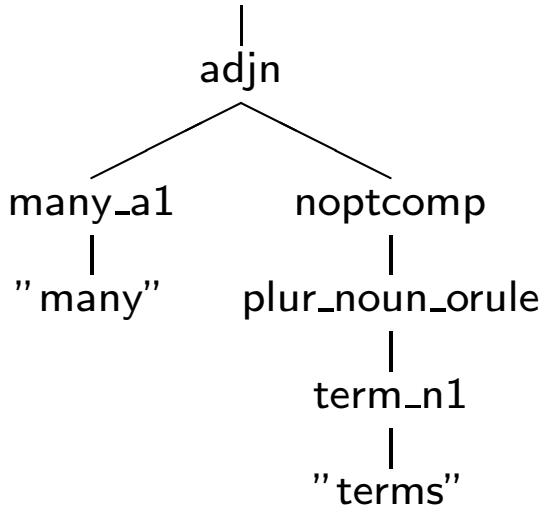
α: ([101,0,1,XCAT,[100]])
β: [('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]])



TRANSITION: 4) PASSIVE (C^2)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

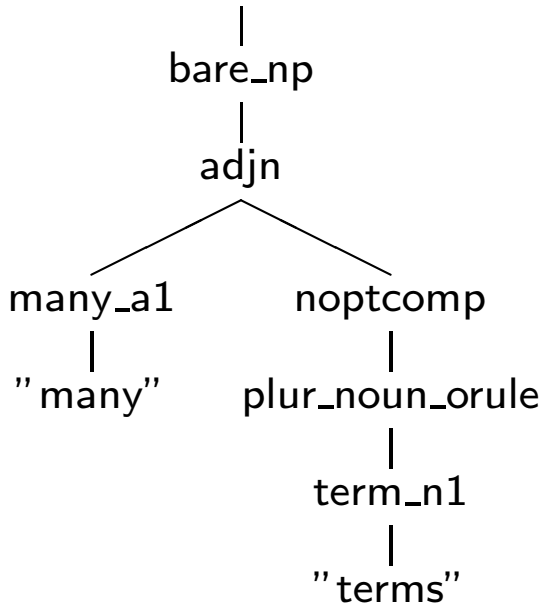
α: ([])
β: [('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]])



TRANSITION: 2) UNIT (C^1)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

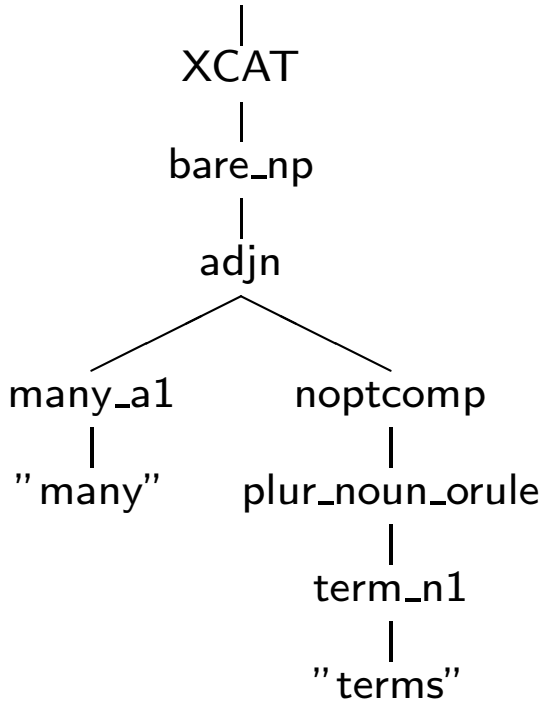
α: ([])
β: [('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]])



TRANSITION: 3) ACTIVE

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

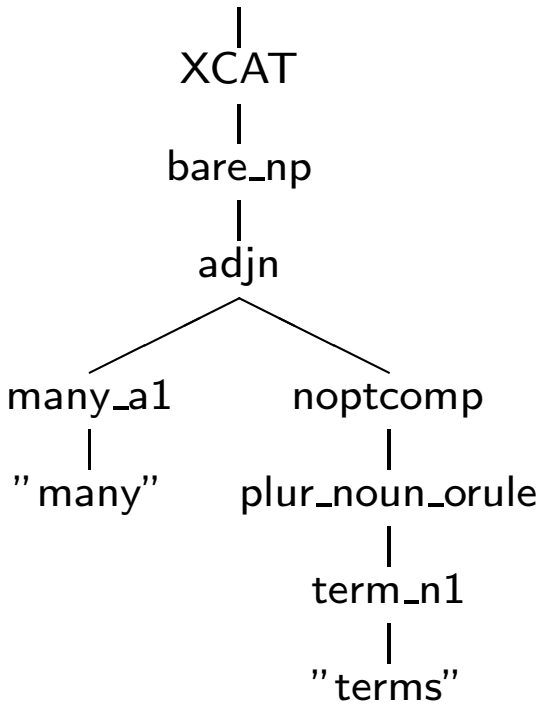
α: ([107,0,2,XCAT,[106]]])
β: [('be_c_are', '"are"', ['2', '3']), ('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]])

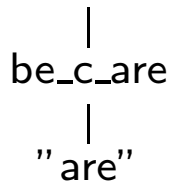


TRANSITION: 1) READ

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

α: ([107,0,2,XCAT,[106]]])
β: [('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]], [108,2,3,be_c_are,[]])

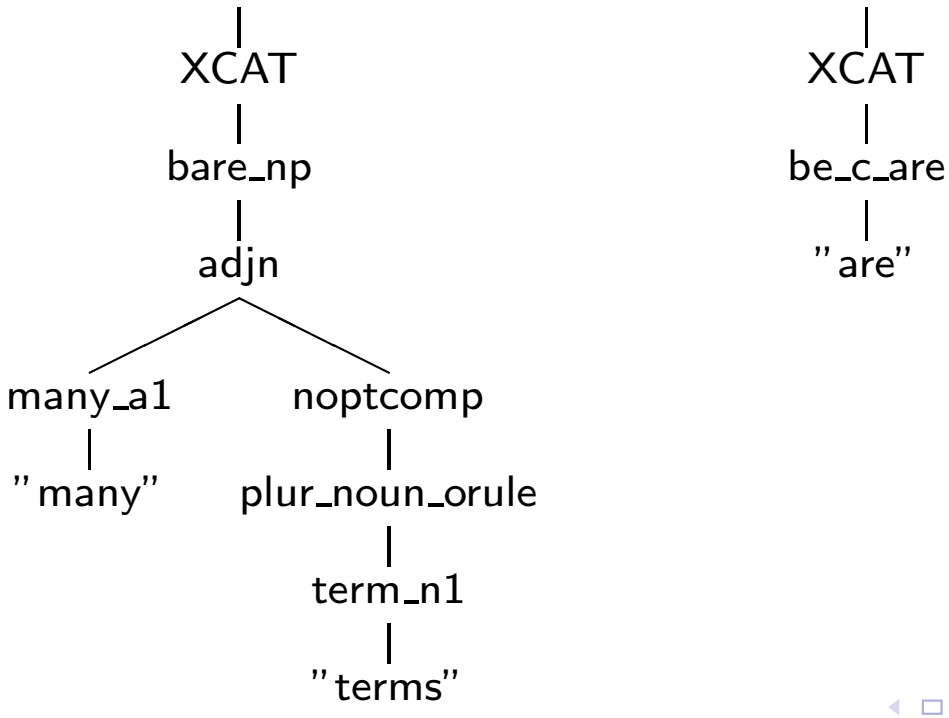




TRANSITION: 3) ACTIVE

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

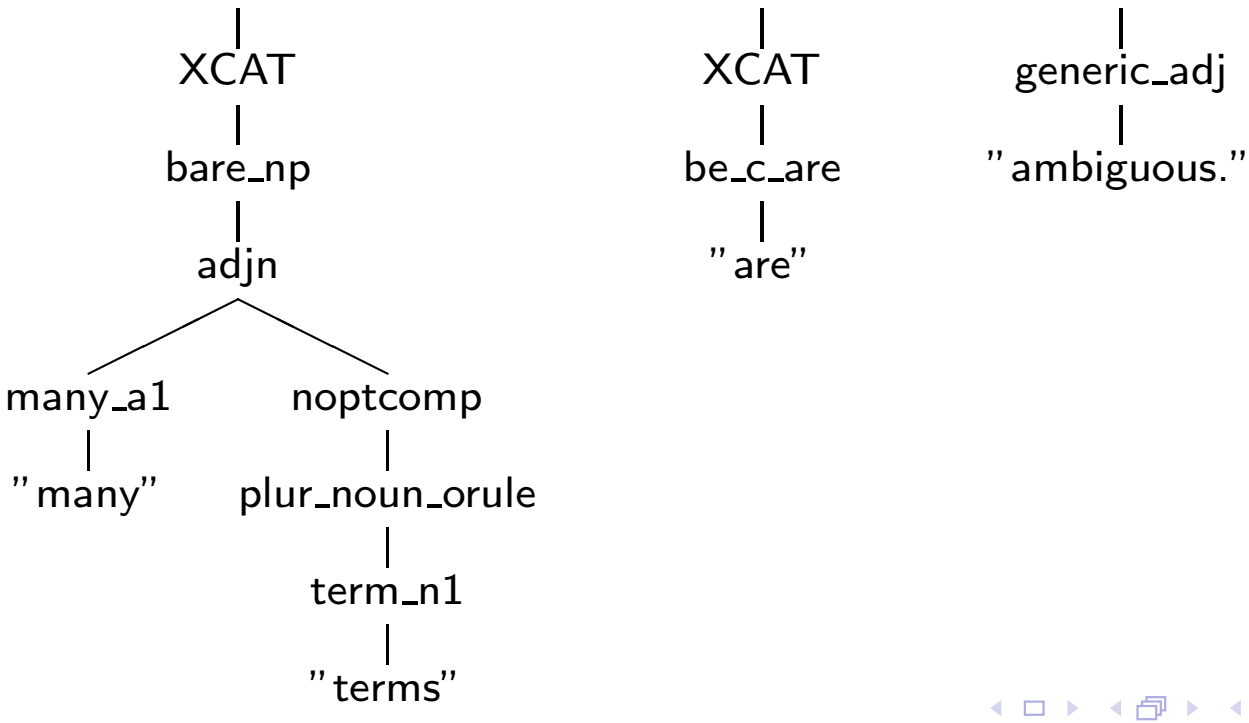
α: ([107,0,2,XCAT,[106]]], [109,2,3,XCAT,[108]])
β: [('generic_adj', '"ambiguous."', ['3', '4'])]
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]], [108,2,3,be_c_are,[]])



TRANSITION: 1) READ

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

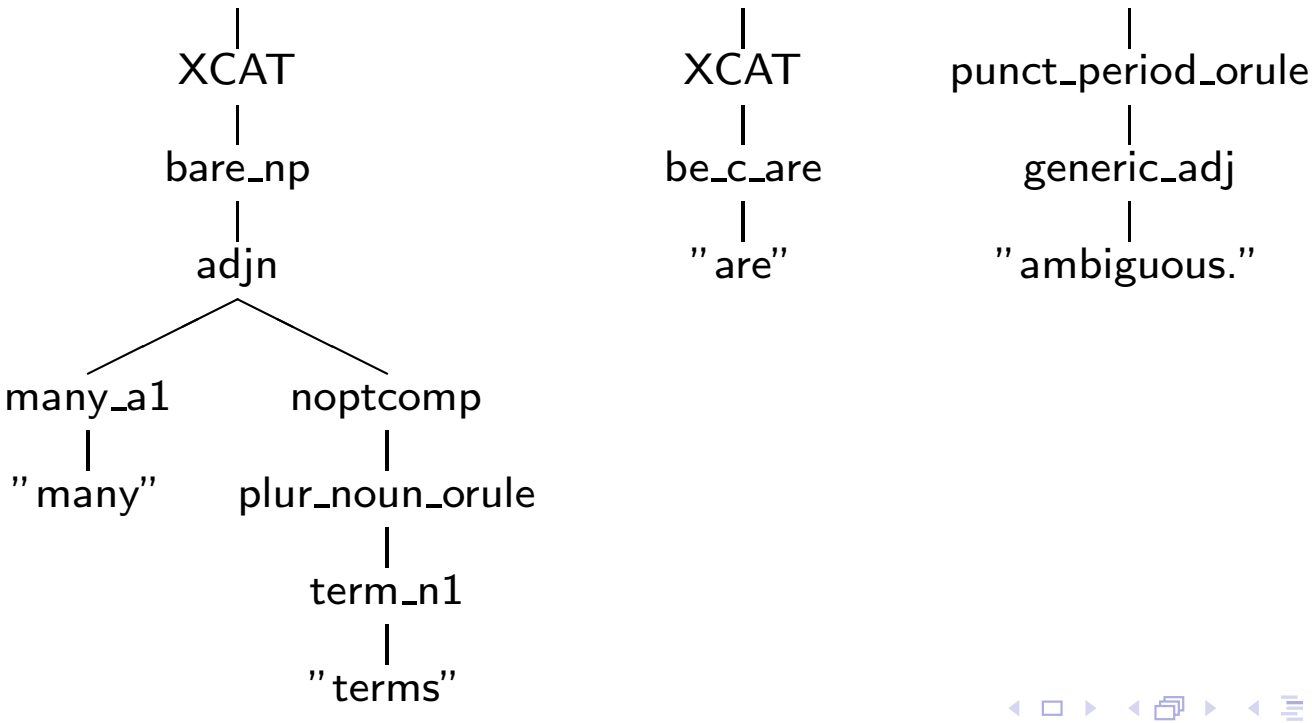
α: ([107,0,2,XCAT,[106]]], [109,2,3,XCAT,[108]])
β: []
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]], [108,2,3,be_c_are,[]], [110,3,4,generic_adj,[]])



TRANSITION: 2) UNIT (C^1)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

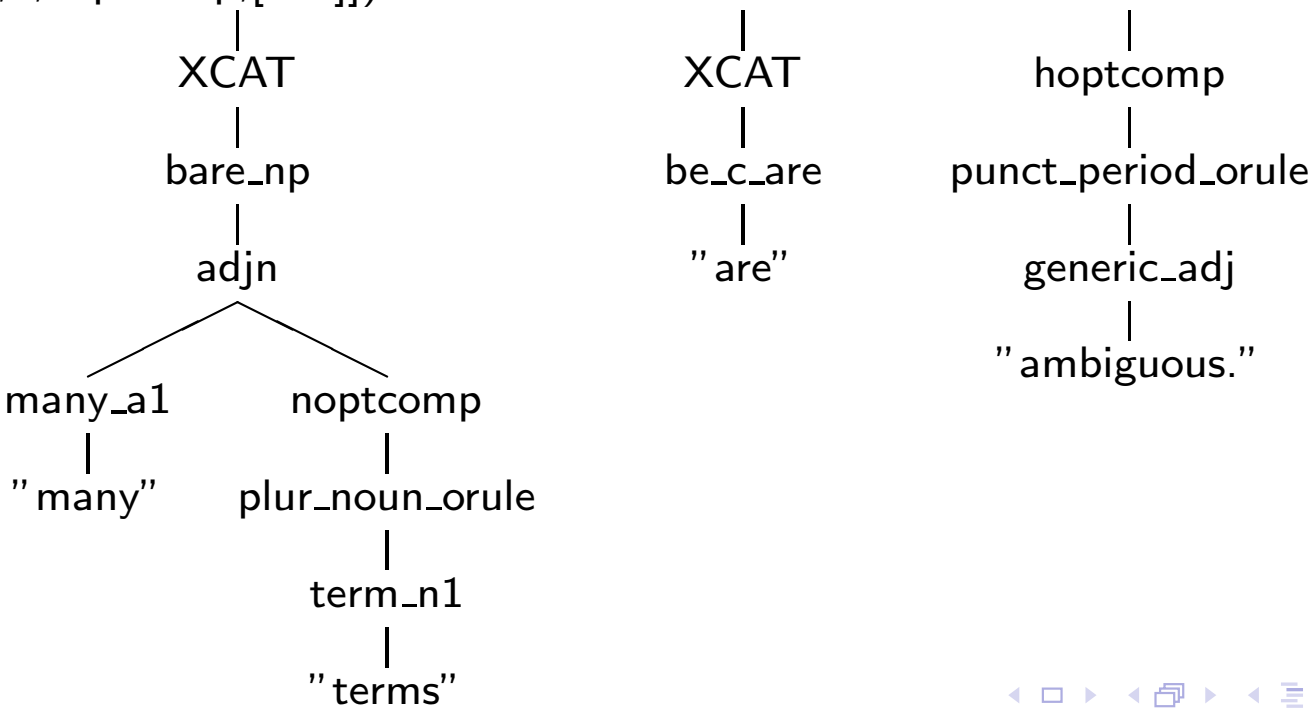
α: ([107,0,2,XCAT,[106]]], [109,2,3,XCAT,[108]])
β: []
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]], [108,2,3,be_c_are,[]], [110,3,4,generic_adj,[]],[111,3,4,punct_period_orule,[110]])



TRANSITION: 2) UNIT (C^1)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

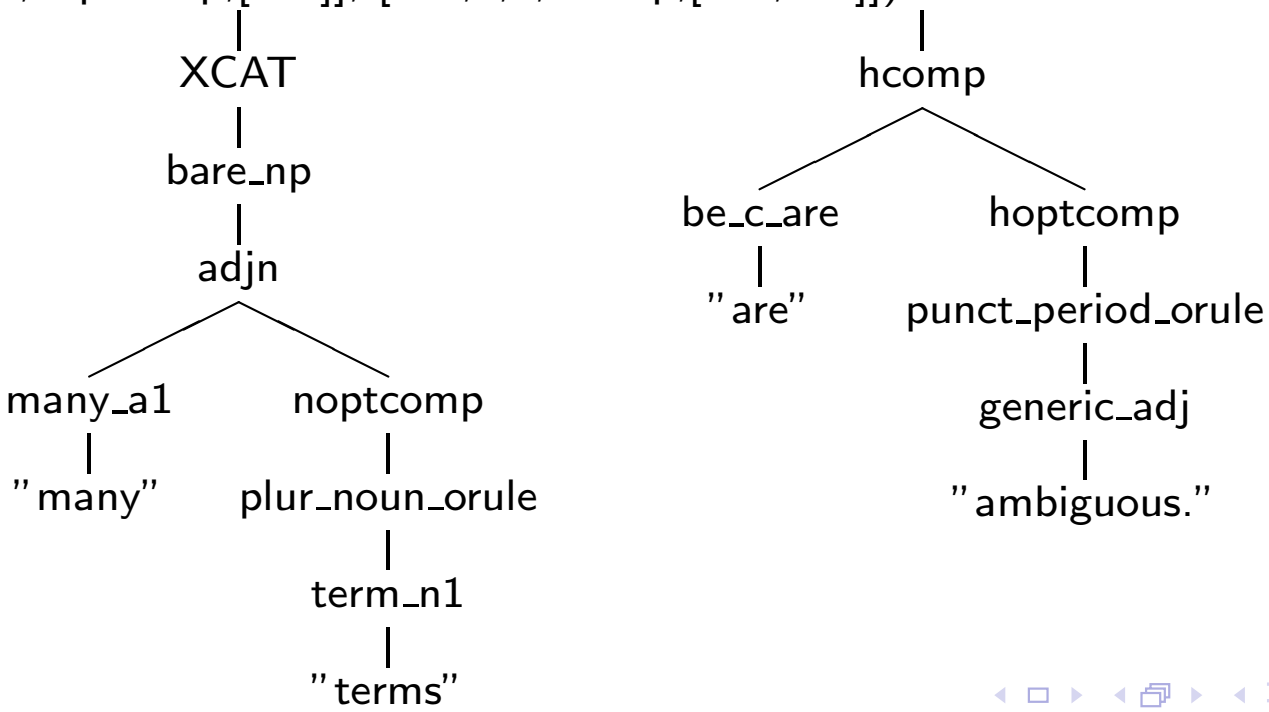
α: ([107,0,2,XCAT,[106]]], [109,2,3,XCAT,[108]])
β: []
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]], [108,2,3,be_c_are,[]], [110,3,4,generic_adj,[]], [111,3,4,punct_period_orule,[110]], [112,3,4,hoptcomp,[111]])



TRANSITION: 4) PASSIVE (C^2)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

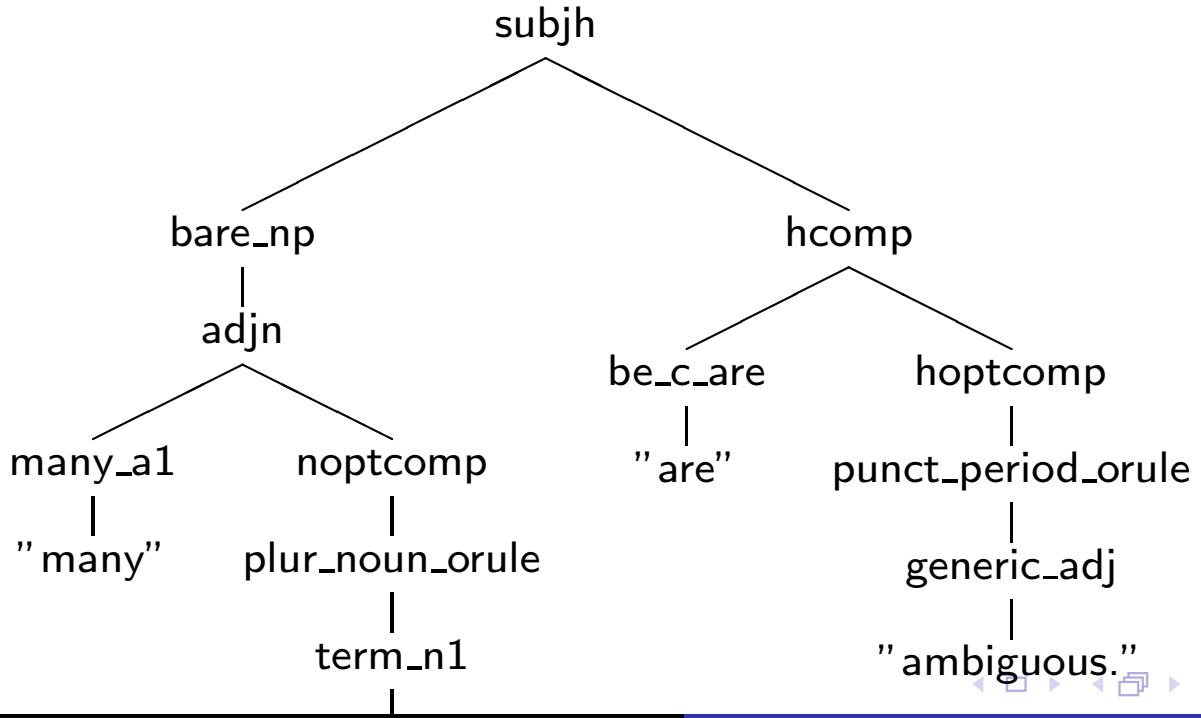
α: ([107,0,2,XCAT,[106]]],)
β: []
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]], [108,2,3,be_c_are,[]], [110,3,4,generic_adj,[]],[111,3,4,punct_period_orule,[110]], [112,3,4,hoptcomp,[111]], [113,2,4,hcomp,[108,112]])



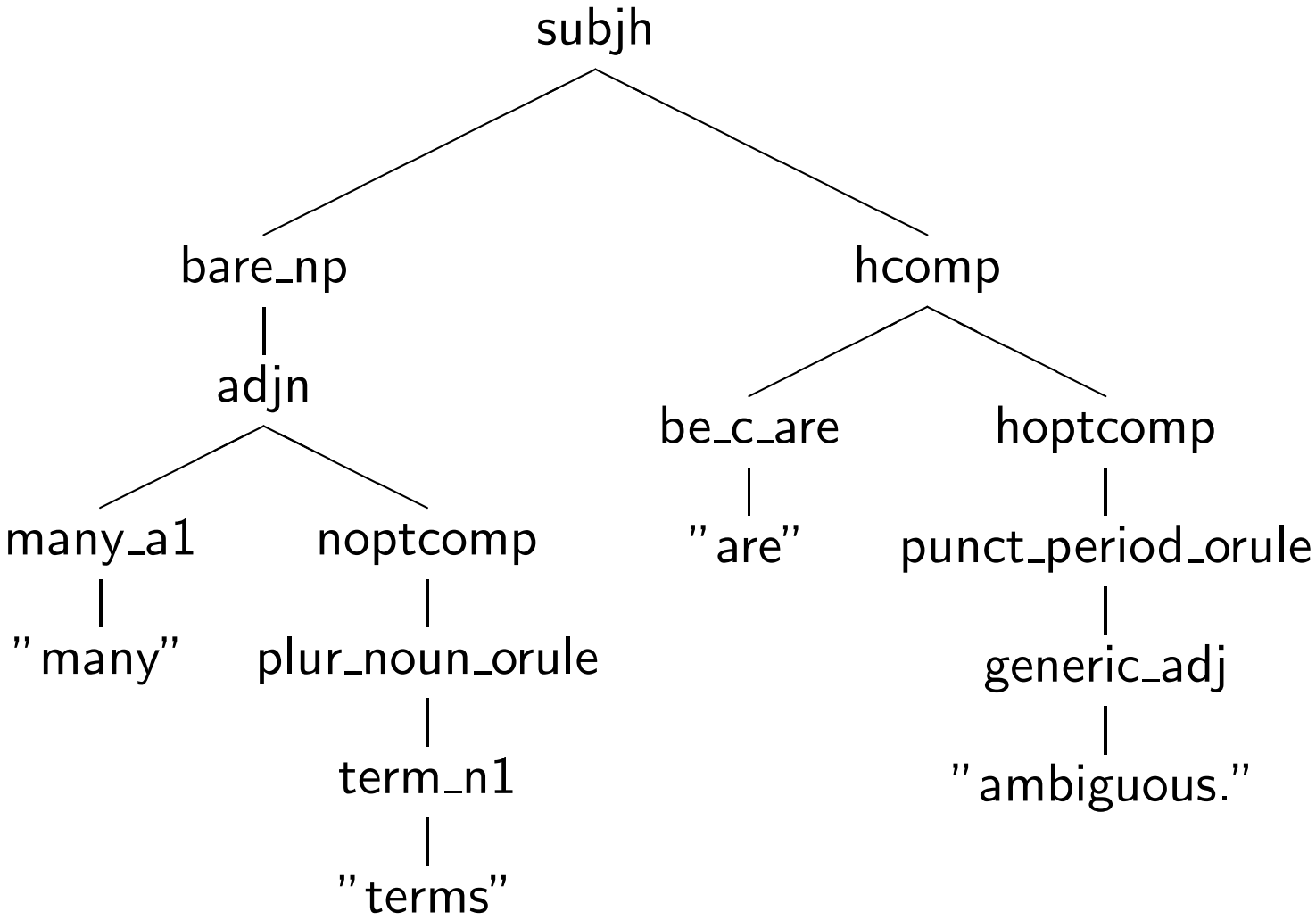
TRANSITION: 4) PASSIVE (C^2)

- 1) READ
- 2) UNIT (C^1)
- 3) ACTIVE
- 4) PASSIVE (C^2)

α: ([])
β: []
π: ([100,0,1,many_a1,[]], [102,1,2,term_n1,[]], [103,1,2,plur_noun_orule,[102]], [104,1,2,noptcomp,[103]], [105,0,2,adjn,[100,104]], [106,0,2,bare_np,[105]], [108,2,3,be_c_are,[]], [110,3,4,generic_adj,[]], [111,3,4,punct_period_orule,[110]], [112,3,4,hoptcomp,[111]], [113,2,4,hcomp,[108,112]], [114,0,4,subjh,[106,113]])



TRANSITION: ACCEPTED



```
(subjh 0 4
  (bare_np 0 2
    (adjn 0 2
      (many_a1 0 1
        ("many" ) )
      (noptcomp 1 2
        (plur_noun_orule 1 2
          (term_n1 1 2
            ("terms" ) ) ) ) ) )
  (hcomp 2 4
    (be_c_are 2 3
      ("are" ) )
    (hoptcomp 3 4
      (punct_period_orule 3 4
        (generic_adj 3 4
          ("ambiguous." ) ) ) ) ) )
10021300 many terms are ambiguous.
```

Oracle

- An *Oracle* picks a transition out of every parse configuration
- The oracle will use:
  1. inherent preconditions (after ACTIVE, READ is the only legal transition)
  2. if no such conditions apply:
     2.1 classification based on machine learning and ERG rules
- A substantial part of the project will be experimentation with the feature model for machine learning and integration of ERG rules.

Feature Model

An example of the feature model for the first gold standard parse transitions for the test sentence:

```
TRANS=unit:bare_np N0=be_c_are N1=generic_adj STACK=nil T0=:adjn T1=treen1nil LAST=TRANS=passive:adjn
TRANS=active N0=be_c_are N1=generic_adj STACK=nil T0=:bare_np T1=treen1nil LAST=TRANS=unit:bare_np
TRANS=read N0=be_c_are N1=generic_adj STACK=0 T0=:bare_np T1=treen1nil LAST=TRANS=active
TRANS=active N0=generic_adj N1=</s> STACK=-2 T0=be_c_are T1=:bare_np LAST=TRANS=read
TRANS=read N0=generic_adj N1=</s> STACK=0 T0=be_c_are T1=:bare_np LAST=TRANS=active
TRANS=unit:punct_period_orule N0=</s> N1=</s> STACK=-1 T0=generic_adj T1=be_c_are LAST=TRANS=read
TRANS=unit:hoptcomp N0=</s> N1=</s> STACK=-1 T0=:punct_period_orule T1=be_c_are LAST=TRANS=unit:punct_peri
TRANS=passive:hcomp N0=</s> N1=</s> STACK=-1 T0=:hoptcomp T1=be_c_are LAST=TRANS=unit:hoptcomp
TRANS=passive:subjh N0=</s> N1=</s> STACK=-2 T0=:hcomp T1=be_c_are LAST=TRANS=passive:hcomp
TRANS=finished N0=</s> N1=</s> STACK=nil T0=:subjh T1=be_c_are LAST=TRANS=passive:subjh
```