Syntax:
Unification-based Grammar

Berthold Crysmann
crysmann@dfki.de

Overview

- **Context-free grammars and NL**
- **Features and Feature Structures**
- **Unification**
- **CFG+Unification**
  - PATR
  - LFG
- **Outlook**

Natural language and Context-Free Grammars (CFG)

- **Minimal grammar type (Chomsky hierarchy) capable of describing natural languages such as English**
  - Assumption: Languages are mere sets of strings
  - Centre self-embedding
- **Not all languages of the world are describable by CFGs**
  - Cross-serial dependencies in Swiss German (NP1 NP2 NP3 V1 V2 V3)
  - Reduplication
- **CFG well equipped to model constituency and precedence relations**
- **Atomic symbols (of CFGs) do not permit to access individual properties of parts-of-speech**
  - Subcategorisation (government)
  - Agreement
- **Phenomena can only be covered extensionally**
  - enumerating all possible combinations of atomic symbols

Source: Berthold Crysmann 2006

Natural language and Context-Free Grammars (CFG)

- **Subcategorisation:**
  - Lexical heads of the same category (e.g., verbs) often differ according to the number of arguments they take
    - *Intransitive*
      John slept.
    - *Transitive*
      John killed the burglar.
    - Ditransitive
      John gave the jewels to the burglar.
  - Category symbols in CFG are atomic labels
    - Distinction of subcategorisation frames can only be modelled by introduction of new

```
VP -> Vi
VP -> Vt NP
VP -> Vd NP PP
```

    - Common properties of VPs unexpressed
      i.e., that they all contain a head of the same basic category (=V)
    - Lexical nature of requirement cannot be captured

Source: Berthold Crysmann 2006

Natural language and Context-Free Grammars (CFG)

- **Agreement:**
  - Syntactic elements often agree according to morpho-syntactic features, e.g., person/number agreement between subject and finite verb
    I am happy, you are happy, he is happy, etc.
  - In CFGs, featural distinction must, again, be encoded as different atomic labels
    e.g., `NP1s, NP2s, NP3s, ... V1s, V2s , ...`
  - Differentiation according to agreement features involves all category symbols and PS rules along the path between nodes in the agreement relation

  - E.g.

```
S-> NP VP           VP -> V AP
```

  - Becomes

```
S-> NP1s VP1s       VP1s -> V1s AP
S-> NP2s VP2s       VP2s -> V2s AP
S-> NP2s VP3s       VP3s -> V3s AP
...
```

Natural language and Context-Free Grammars (CFG)

- **Long distance dependencies:**
  - Syntactic constituents may undergo extraction, separating them from the heads that govern them
    - Wh-questions
      What do you think John bought [e]?
    - Topicalisation
      It was an ice-cream that John bought [e]?
  - Long distance dependencies can cross (multiple) sentence boundaries
  - Bounded numbers of long distance dependencies can be encoded with CFGs
    - Locally missing constituents are encoded as part of the category symbol of every node along the extraction path
    - E.g., to derive *A book, he bought.*

```
S-> NP VP
VP -> V NP
```

    - Must be augmented by

```
VP/NP -> V
S/NP-> NP VP/NP
S -> NP S/NP
```

Source: Berthold Crysmann 2006

Natural language and Context-Free Grammars (CFG)

- **Coordination:**
  - In many natural languages, only like categories can be conjoined (=combined with conjunctions such as *and* or *or*)
  - Examples
    - *Sentential coordination*
      [[Jack fell down and broke his crown] and [Jill came tumbling after]].
    - *VP coordination*
      Jack [[fell down] and [broke his crown]].
    - *NP coordination*
      [[Jack] and [Jill]] went up the hill
    - AP, PP coordination
    - Coordination at lexical level (N, A, P, V)
  - CFGs cannot state likeness of category as such, but have to enumerate all and every combination of compatible symbols
    - E.g.

```
S -> S Conj S
VP -> VP Conj VP
NP -> NP Conj NP
...
```

Source: Berthold Crysmann 2006

Feature Structures

- Idea: Sets of (linguistic) objects can be described by their properties
- Properties can be represented as attribute-value pairs (=features)
- Values can be atomic or complex (feature structures)
- Examples:
  - Categorial information

$$\begin{bmatrix} \text{cat:} & \text{v} \end{bmatrix}$$

  - Agreement information

$$\begin{bmatrix} \text{per:} & 3 \\ \text{num:} & \text{sg} \end{bmatrix}$$

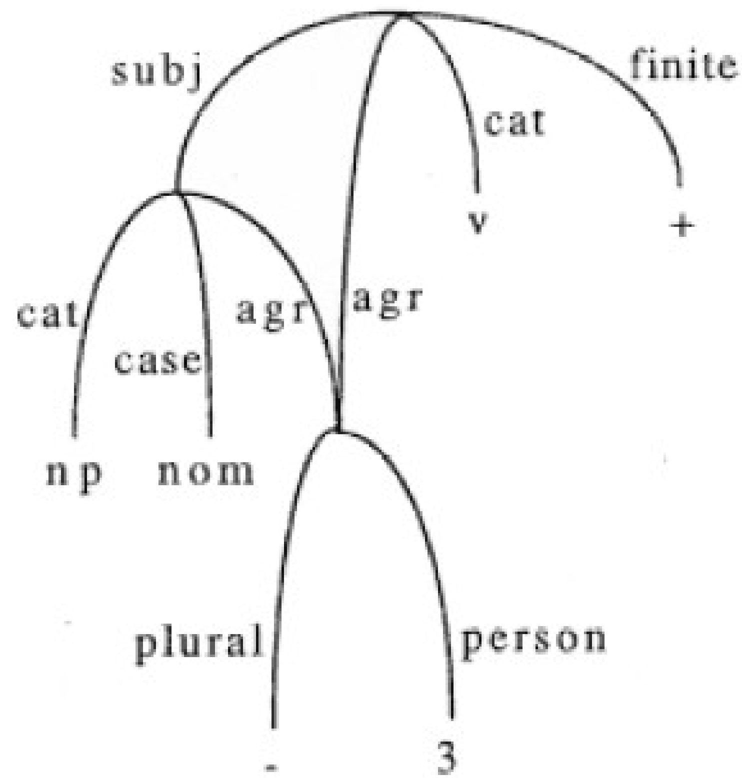
  - Verb with agreement information

$$\begin{bmatrix} \text{cat:} & \text{v} \\ \text{agr:} & \begin{bmatrix} \text{per:} & 3 \\ \text{num:} & \text{sg} \end{bmatrix} \end{bmatrix}$$

Source: Berthold Crysmann 2006

Feature Structures – Reentrancies

- Feature structures can be represented as directed (acyclic) graphs (DAGs)
- Paths in a feature structure graph can share a value (structure sharing)
- Structure sharing is a powerful tool to express necessary identity of values



Source: Berthold Crysmann 2006

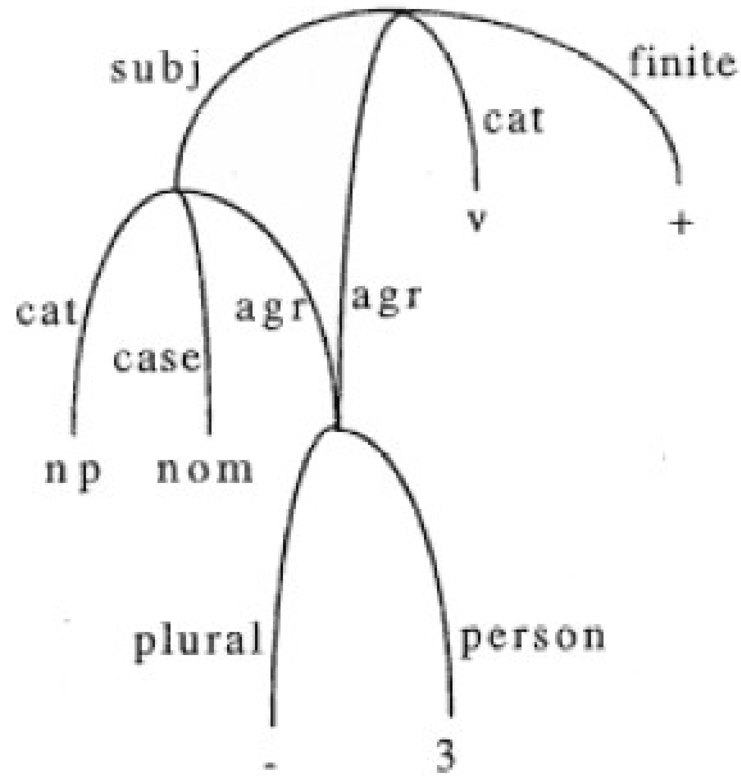
Feature Structures – Representation formats

- **Feature structures graphs can be described by**
  - Path equations

```
<cat> = v
<finite> = +
<agr:plural> = -
<agr:person> = 3
<subj:cat> = np
<subj:case> = nom
<subj:agr> = <agr>
```

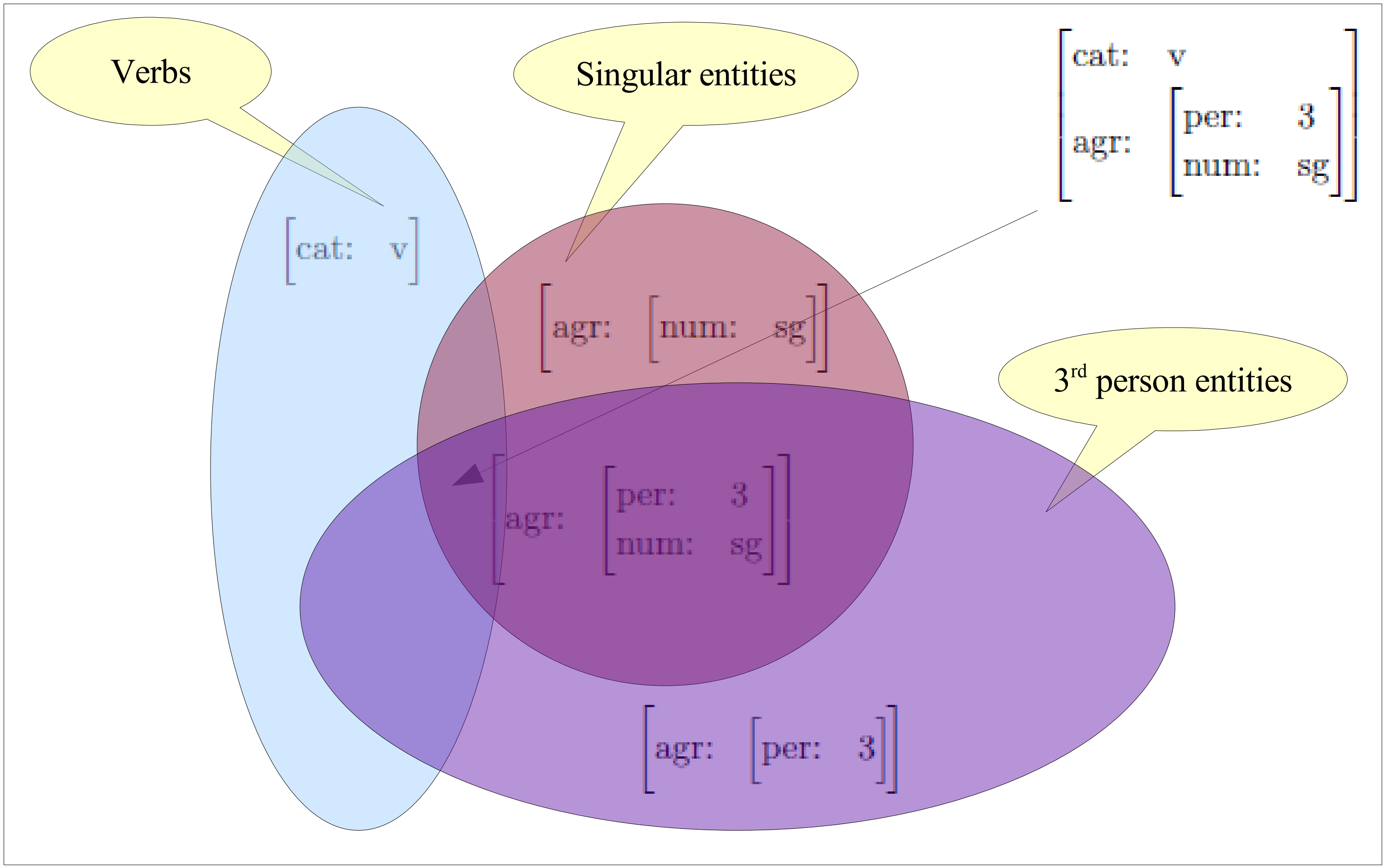
  - Attribute-value matrices

$$\begin{bmatrix} \text{cat:} & \text{v} \\ \text{finite:} & + \\ \text{agr:} & \boxed{1}\begin{bmatrix} \text{plural:} & - \\ \text{per:} & 3 \end{bmatrix} \\ \text{subj:} & \begin{bmatrix} \text{cat:} & \text{np} \\ \text{case:} & \text{nom} \\ \text{agr:} & \boxed{1}\begin{bmatrix} \ \end{bmatrix} \end{bmatrix} \end{bmatrix}$$



Source: Berthold Crysmanr

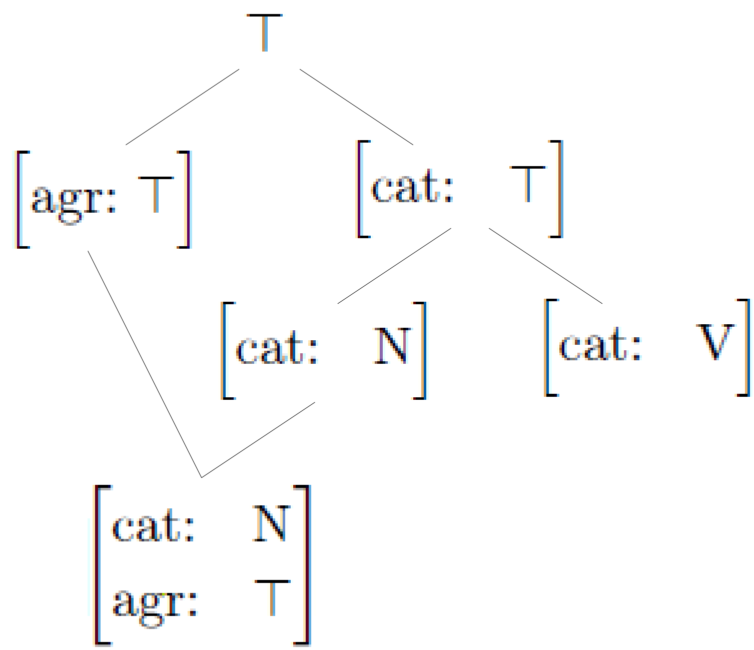
Feature Structures – Denotation



Source: Berthold Crysmann 2006

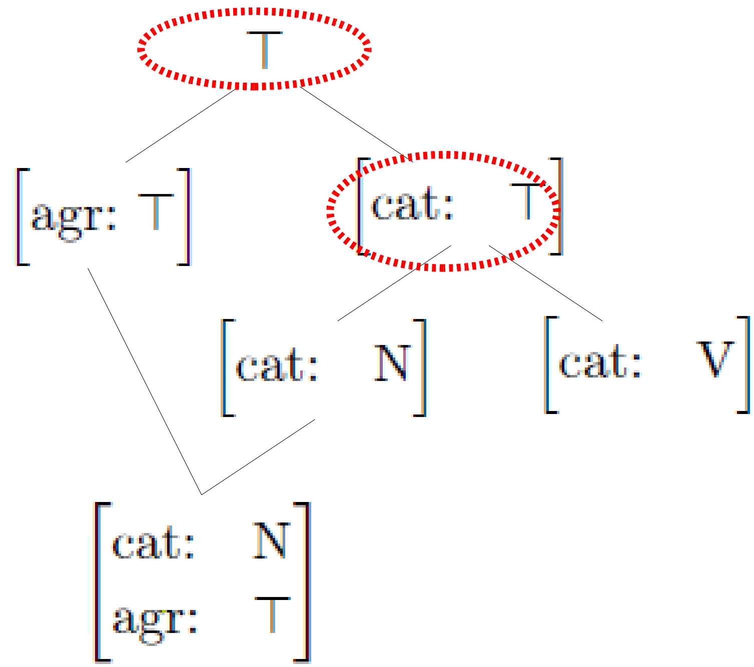
Subsumption

- Sets of feature structure terms can be ordered according to the amount of information they encode
- Less informative feature structures are said to subsume more informative ones
- Subsumption relation is
  - Reflexive
  - Antisymmetric
  - Transitive
- Relevant information consists of
  - Paths
  - Values
  - Reentrancies (path equations)



Subsumption

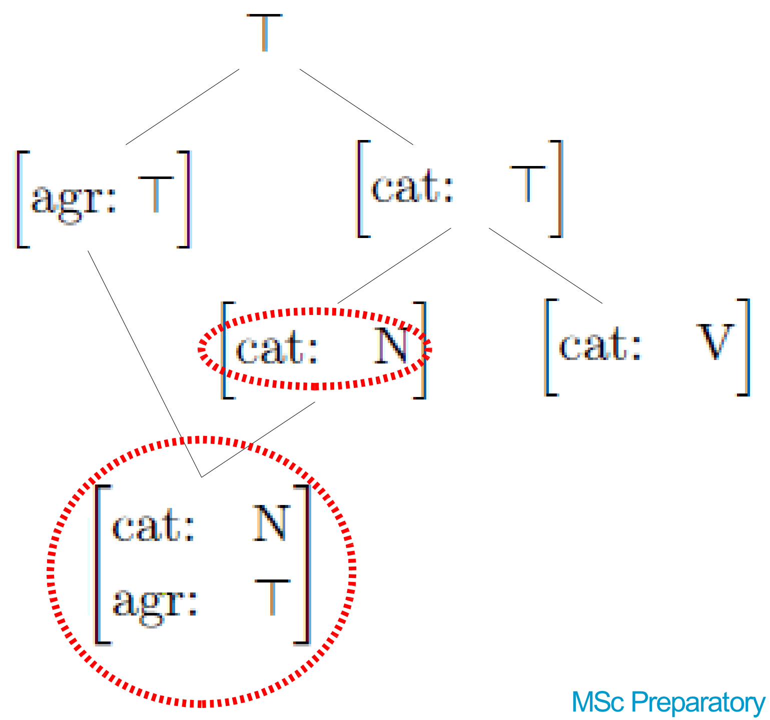
- Sets of feature structure terms can be ordered according to the amount of information they encode
- Less informative feature structures are said to subsume more informative ones
- Subsumption relation is
  - Reflexive
  - Antisymmetric
  - Transitive
- Relevant information consists of
  - Paths
  - Values
  - Reentrancies (path equations)



Source: Berthold Crysmann 2006

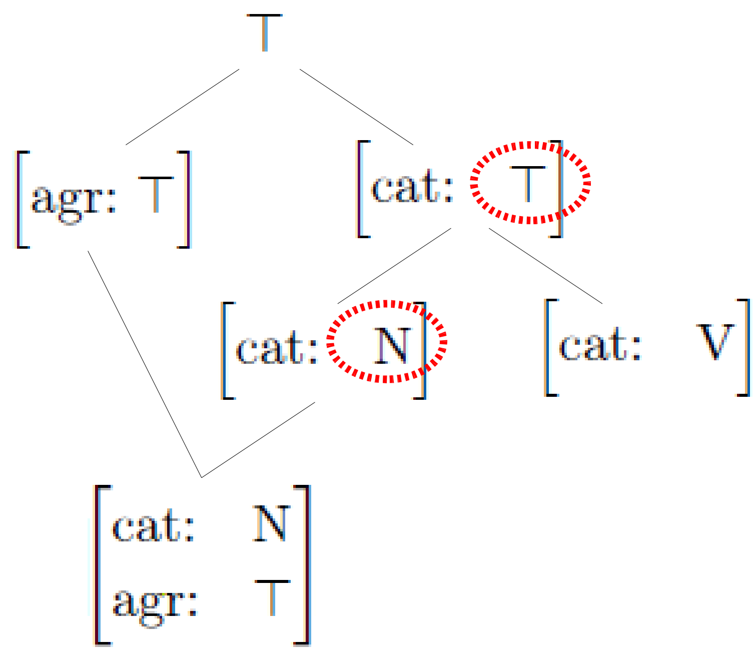
Subsumption

- Sets of feature structure terms can be ordered according to the amount of information they encode
- Less informative feature structures are said to subsume more informative ones
- Subsumption relation is
  - Reflexive
  - Antisymmetric
  - Transitive
- Relevant information consists of
  - Paths
  - Values
  - Reentrancies (path equations)



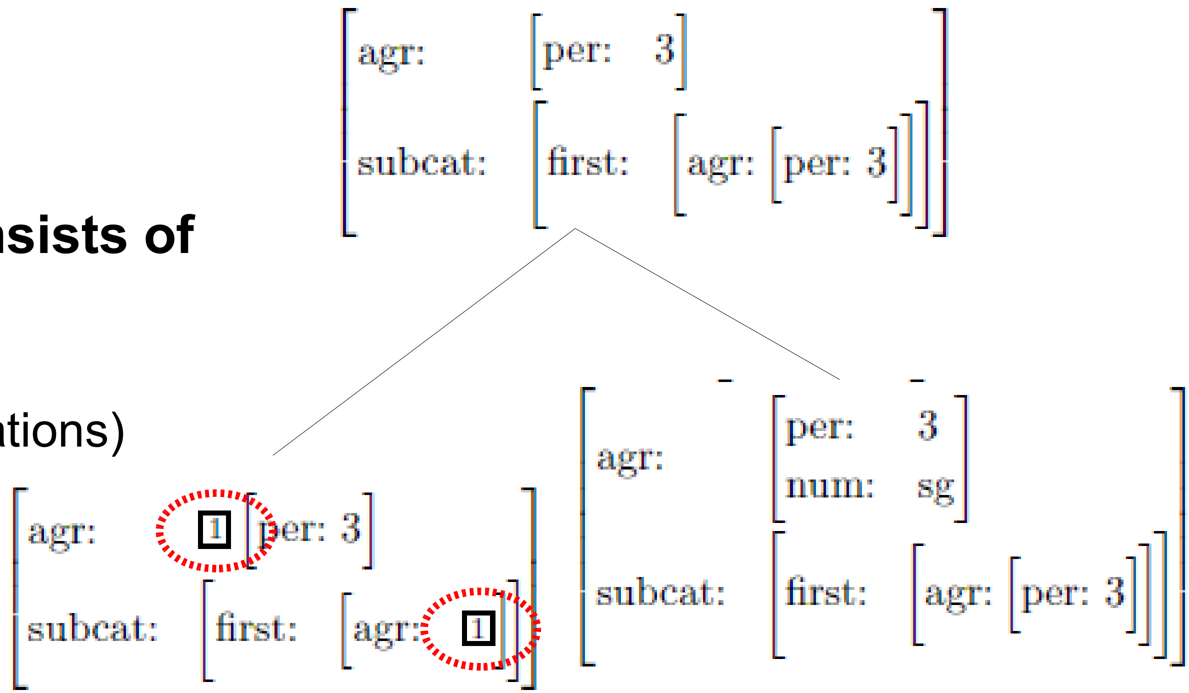
Subsumption

- Sets of feature structure terms can be ordered according to the amount of information they encode
- Less informative feature structures are said to subsume more informative ones
- Subsumption relation is
  - Reflexive
  - Antisymmetric
  - Transitive
- Relevant information consists of
  - Paths
  - Values
  - Reentrancies (path equations)



Subsumption

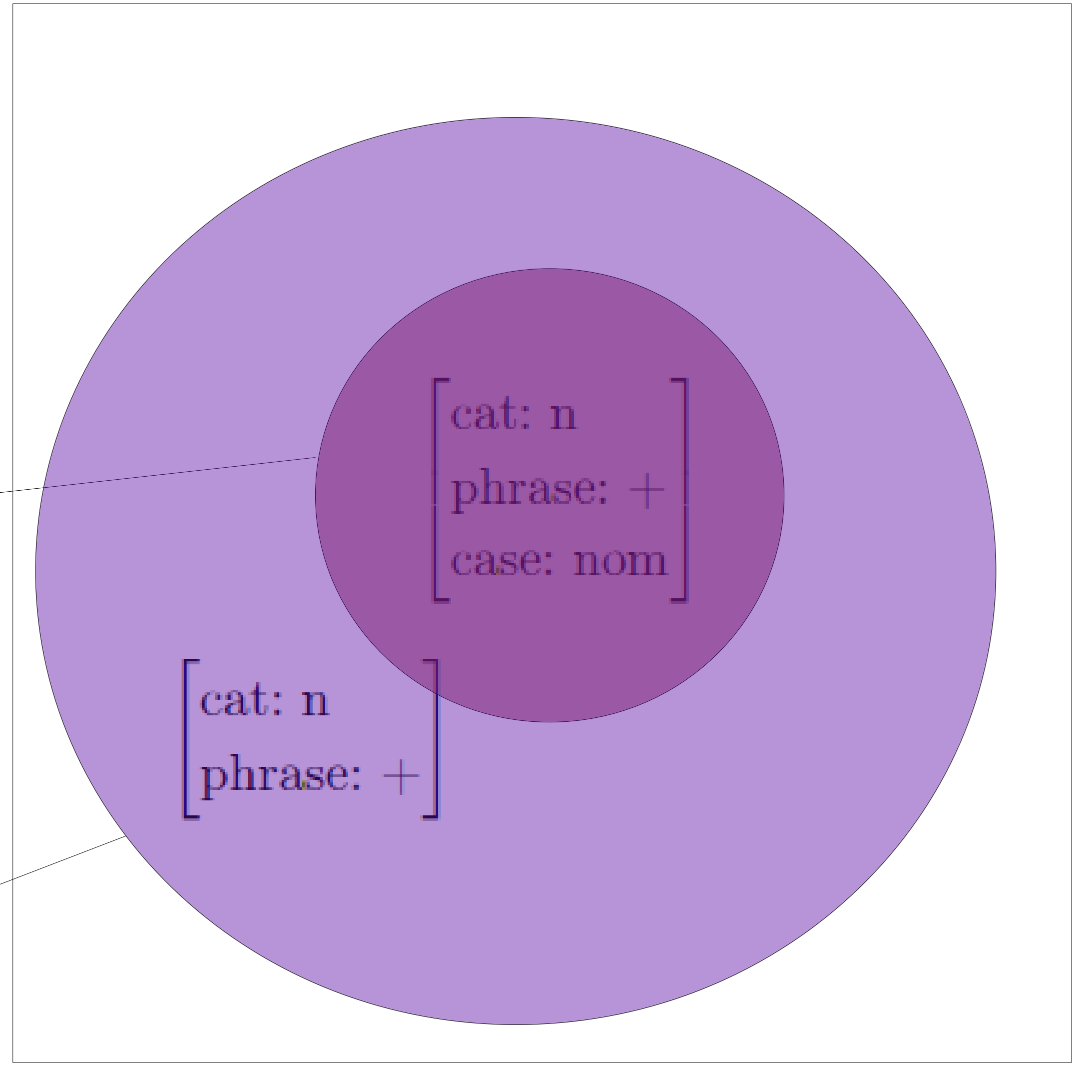
- Sets of feature structure terms can be ordered according to the amount of information they encode
- Less informative feature structures are said to subsume more informative ones
- Subsumption relation is
  - Reflexive
  - Antisymmetric
  - Transitive
- Relevant information consists of
  - Paths
  - Values
  - Reentrancies (path equations)



Semantics of Subsumption

$$\mathrm{t}_1 \sqsubseteq \mathrm{t}_2 \leftrightarrow [[t_2]] \subseteq [[t_1]]$$

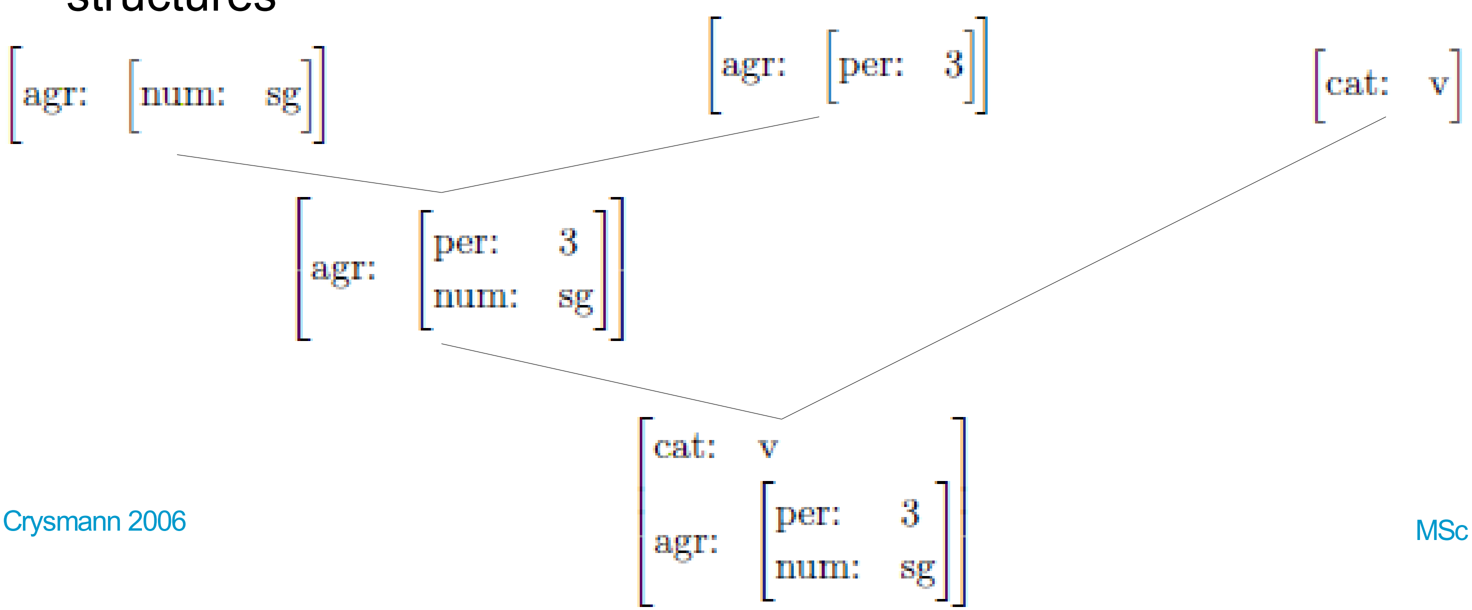
$$\begin{bmatrix} \text{cat: n} \\ \text{phrase: } + \end{bmatrix} \sqsubseteq \begin{bmatrix} \text{cat: n} \\ \text{phrase: } + \\ \text{case: nom} \end{bmatrix}$$



Source: Berthold Crysmann 2006

Unification

- **Characterisation**
  - Operation that combines two feature structures into a new feature structure that contains exactly the information contained in the original feature structure
- **Unification corresponds to**
  - the union of information excluding conflicting information
  - the intersection of sets denoted by the original feature structures
- **Unification defined on the basis of subsumption:**
  - Most general feature structure that is subsumed by both original feature structures

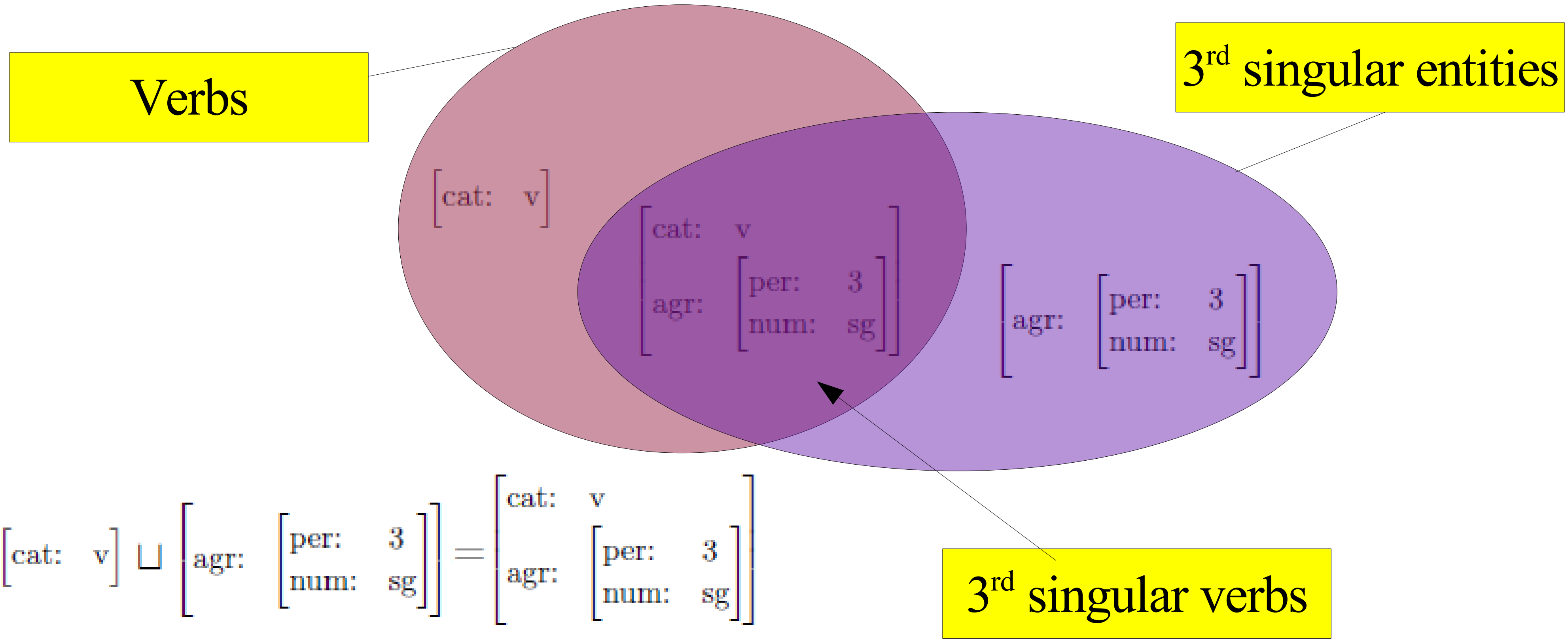


Source: Berthold Crysmann 2006

Semantics of Unification

- Unification of information corresponds to intersection of denotations

$$t_1 \sqcup t_2 \leftrightarrow [[t_1]] \cap [[t_2]]$$



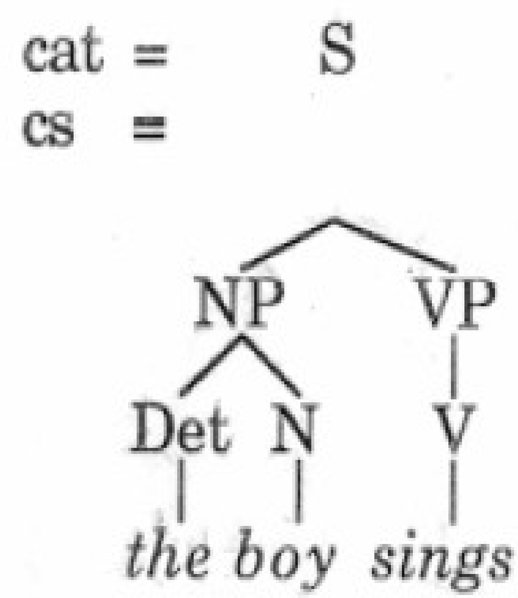
$$\begin{bmatrix}\text{cat:} & \text{v}\end{bmatrix} \sqcup \begin{bmatrix}\text{agr:} & \begin{bmatrix}\text{per:} & 3 \\ \text{num:} & \text{sg}\end{bmatrix}\end{bmatrix} = \begin{bmatrix}\text{cat:} & \text{v} \\ \text{agr:} & \begin{bmatrix}\text{per:} & 3 \\ \text{num:} & \text{sg}\end{bmatrix}\end{bmatrix}$$

Source: Berthold Crysmann 2006

Unification-based grammar (UBG)

- Idea: Combine CFGs with feature structures
- A syntactic entity may be represented as an ordered pair <cat,cs> of a node label (cat) and a constituent structure (cs)

```
<S,
   (<NP,
      (<Det,
         (<the, Ø>)>,
       <N,
         (<boy, Ø>)>)>,
    <VP,
      (<V,
         (<sings, Ø>)>)>)>
```



Source: Berthold Crysmann 2006

Unification-based grammar (UBG)

- ❏ **Simple unification-based grammars replace category label with a feature structure**
- ❏ **Example: PATR-II (Shieber et al.)**

$$S \rightarrow NP \;\; VP \qquad\qquad X_0 \rightarrow X_1 \;\; X_2$$

$$\begin{bmatrix} S: \begin{bmatrix} \text{cat: S} \\ \text{finite:}^{\langle 1 \rangle} \end{bmatrix} \\ NP: \begin{bmatrix} \text{cat: NP} \\ \text{agr:}^{\langle 2 \rangle} \end{bmatrix} \\ VP: \begin{bmatrix} \text{cat: VP} \\ \text{agr:}^{\langle 2 \rangle} \\ \text{finite:}^{\langle 1 \rangle} \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} X_0: \begin{bmatrix} \text{cat: S} \\ \text{finite:}^{\langle 1 \rangle} \end{bmatrix} \\ X_1: \begin{bmatrix} \text{cat: NP} \\ \text{agr:}^{\langle 2 \rangle} \end{bmatrix} \\ X_2: \begin{bmatrix} \text{cat: VP} \\ \text{agr:}^{\langle 2 \rangle} \\ \text{finite:}^{\langle 1 \rangle} \end{bmatrix} \end{bmatrix}$$

Source: Berthold Crysmann 2006

Unification-based grammar (UBG): PATR-II

- **Original notation:**
  - context free rules with node variables
  - Feature structures represented as path equations

```
X0 -> X1 X2
<X0:cat> = S
<X1:cat> = NP
<X2:cat> = VP
<X1:agr> = <X2:agr>
<X0:finite> = <X2:finite>
```

$$X_0 \rightarrow X_1 \quad X_2$$

- **Grammars consists of**
  - Phrase structure rules
  - Lexical entries

$$\begin{bmatrix} X_0: \begin{bmatrix} \text{cat: S} \\ \text{finite:}^{\langle 1\rangle} \end{bmatrix} \\ X_1: \begin{bmatrix} \text{cat: NP} \\ \text{agr:}^{\langle 2\rangle} \end{bmatrix} \\ X_2: \begin{bmatrix} \text{cat: VP} \\ \text{agr:}^{\langle 2\rangle} \\ \text{finite:}^{\langle 1\rangle} \end{bmatrix} \end{bmatrix}$$

Source: Berthold Crysmann 2006

PATR-II example grammar

- **Lexicon:**

some :=

$$\begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

every :=

$$\begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \\ \text{num: sg} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

boy :=

$$\begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \\ \text{num: sg} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

toys :=

$$\begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \\ \text{num: pl} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- **PS rule:**

$$\text{X0} \rightarrow \text{X1 X2 (NP} \rightarrow \text{Det N)}$$

$$\begin{bmatrix} \text{X0:} & \begin{bmatrix} \text{cat:} & \text{NP} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X1:} & \begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X2:} & \begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \boxed{0} \end{bmatrix} \end{bmatrix}$$

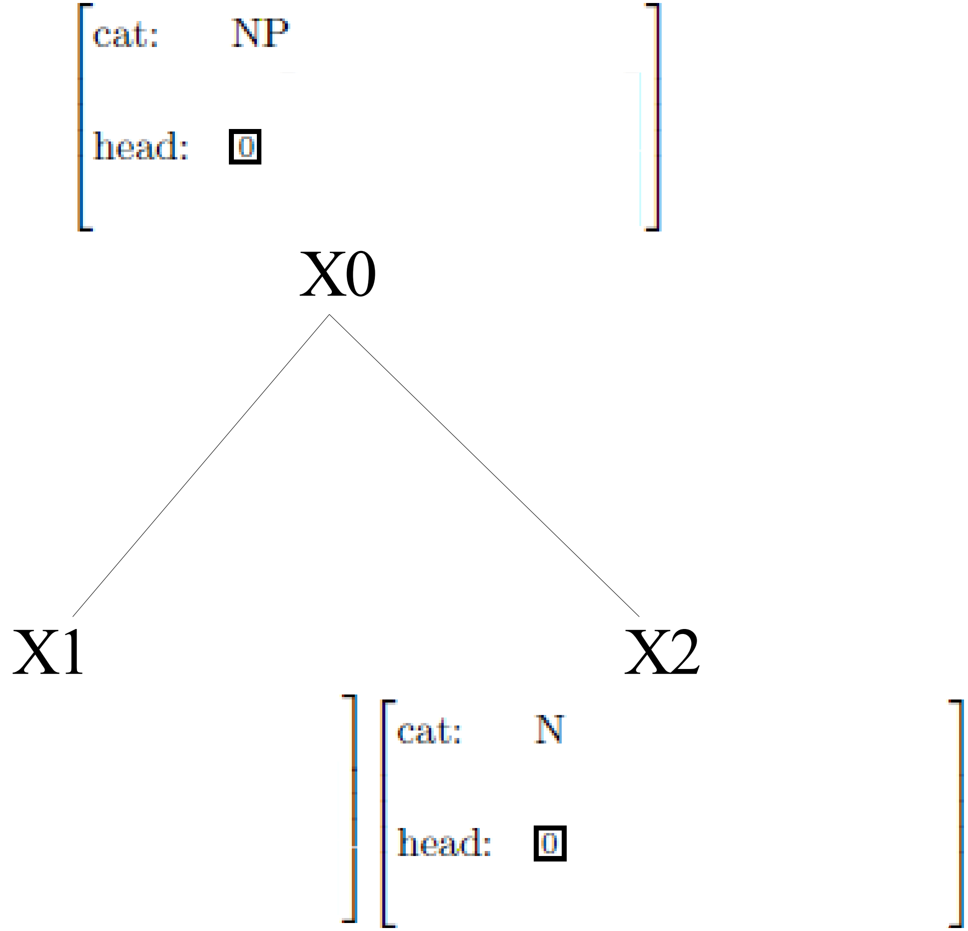
Source: Berthold Crysmann 2006

PATR-II example grammar (*some boy*)

- **Lexicon:**

some :=
[cat: Det
 head: [agr: [per: 3]]]

boy :=
[cat: N
 head: [agr: [per: 3
              num: sg]]]



- **PS rule:**

X0 → X1 X2 (NP → Det N)

[X0: [cat: NP
      head: 0]
 X1: [cat: Det
      head: 0]
 X2: [cat: N
      head: 0]]

PATR-II example grammar

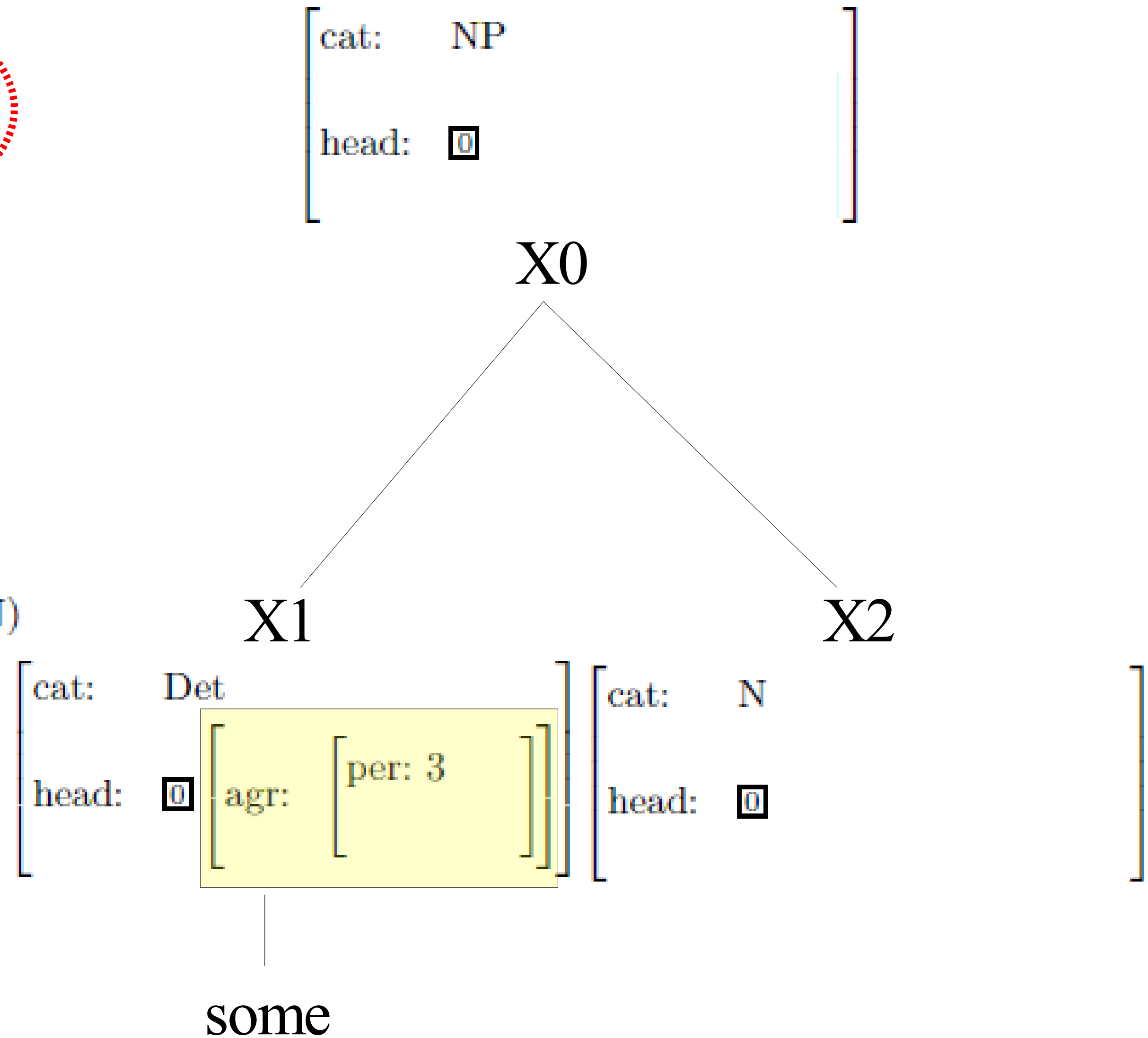
- **Lexicon:**

some :=
$$\begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

boy :=
$$\begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \\ \text{num: sg} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- **PS rule:**

X0 → X1 X2 (NP → Det N)
$$\begin{bmatrix} \text{X0:} & \begin{bmatrix} \text{cat:} & \text{NP} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X1:} & \begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X2:} & \begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \boxed{0} \end{bmatrix} \end{bmatrix}$$



PATR-II example grammar

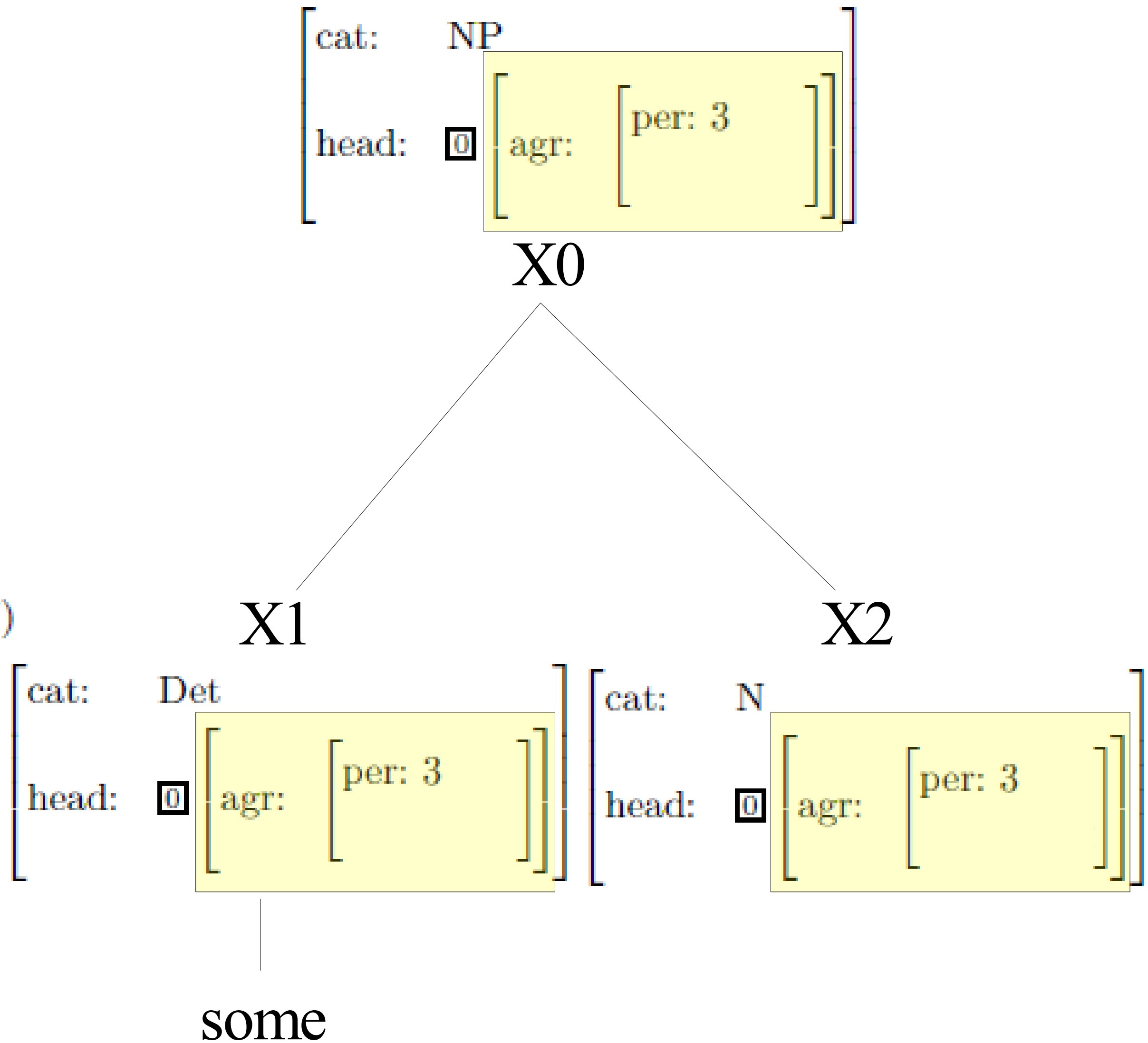
- **Lexicon:**

some :=
$$\begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

boy :=
$$\begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \\ \text{num: sg} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- **PS rule:**

X0 → X1 X2 (NP → Det N)
$$\begin{bmatrix} \text{X0:} & \begin{bmatrix} \text{cat:} & \text{NP} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X1:} & \begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X2:} & \begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \boxed{0} \end{bmatrix} \end{bmatrix}$$



PATR-II example grammar

- **Lexicon:**

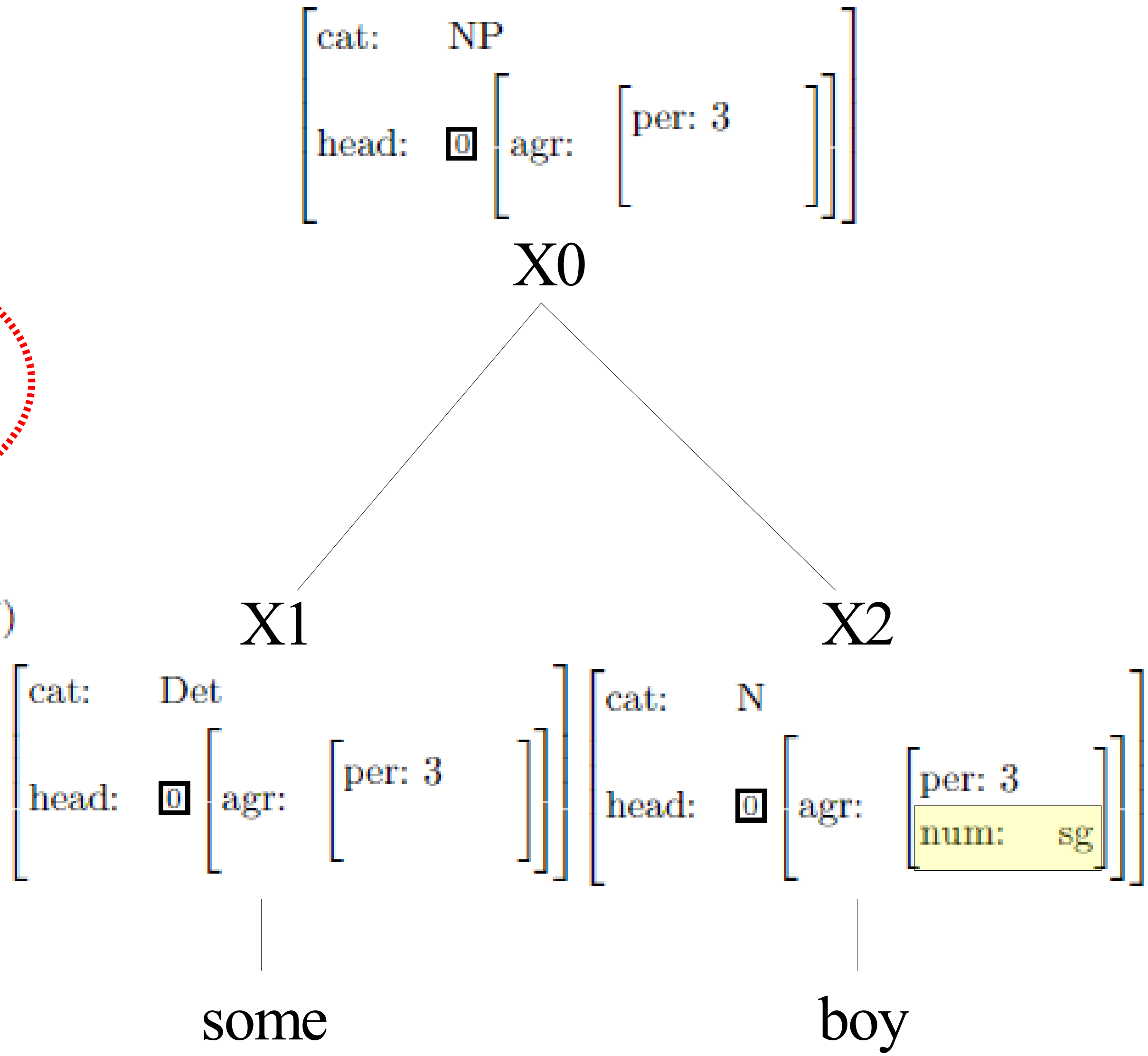
some :=
[cat: Det
head: [agr: [per: 3]]]

boy :=
[cat: N
head: [agr: [per: 3
num: sg]]]

- **PS rule:**

X0 → X1 X2 (NP → Det N)

[X0: [cat: NP
head: [0]]
X1: [cat: Det
head: [0]]
X2: [cat: N
head: [0]]]



Source: Berthold Crysmann 2006

PATR-II example grammar

- **Lexicon:**

some :=

$$\begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

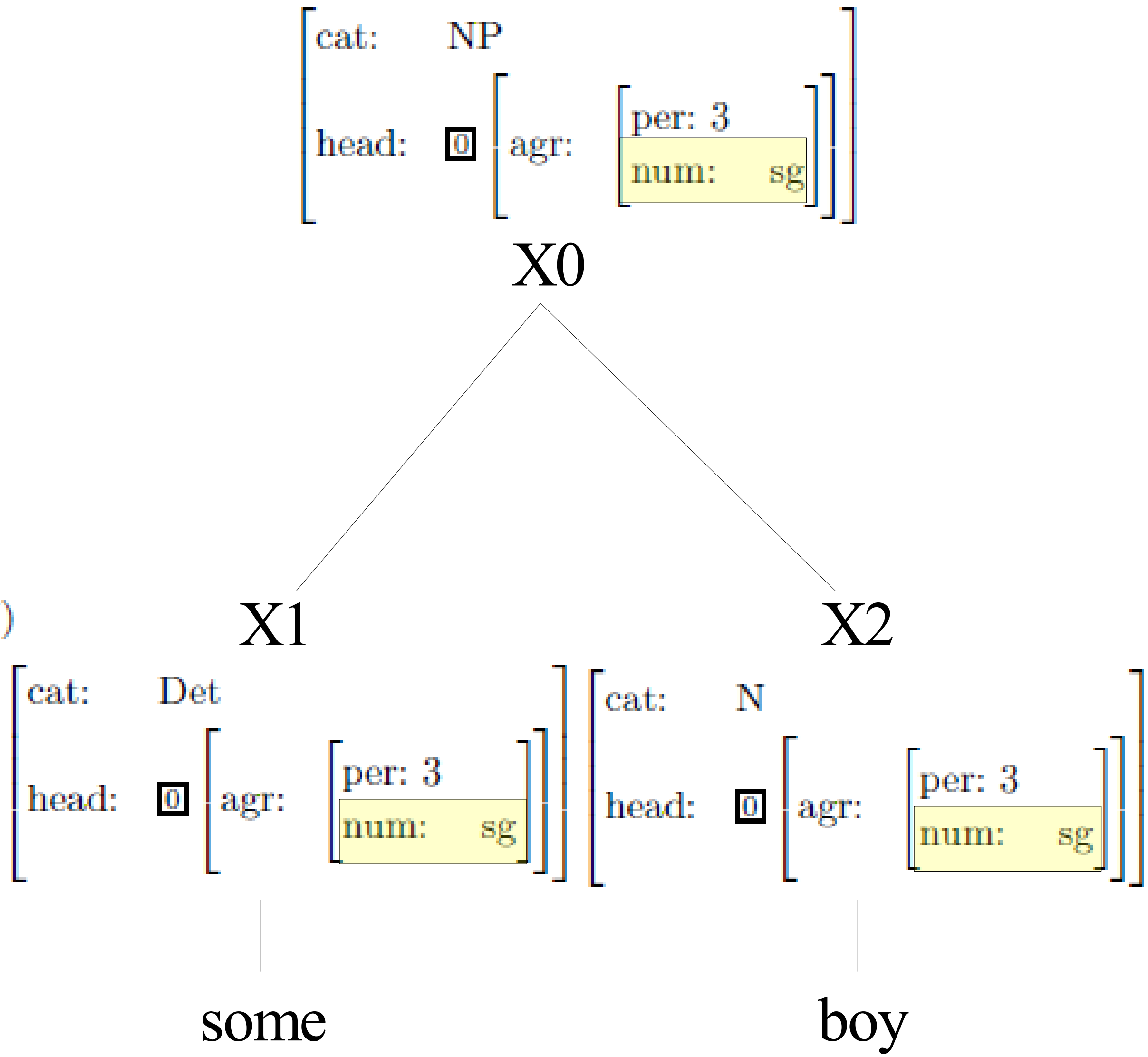
boy :=

$$\begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per: } 3 \\ \text{num: sg} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- **PS rule:**

X0 → X1 X2 (NP → Det N)

$$\begin{bmatrix} \text{X0:} & \begin{bmatrix} \text{cat:} & \text{NP} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X1:} & \begin{bmatrix} \text{cat:} & \text{Det} \\ \text{head:} & \boxed{0} \end{bmatrix} \\ \text{X2:} & \begin{bmatrix} \text{cat:} & \text{N} \\ \text{head:} & \boxed{0} \end{bmatrix} \end{bmatrix}$$



Source: Berthold Crysmann 2006

PATR-II example grammar (**_*every toys_**)

- **Lexicon:**

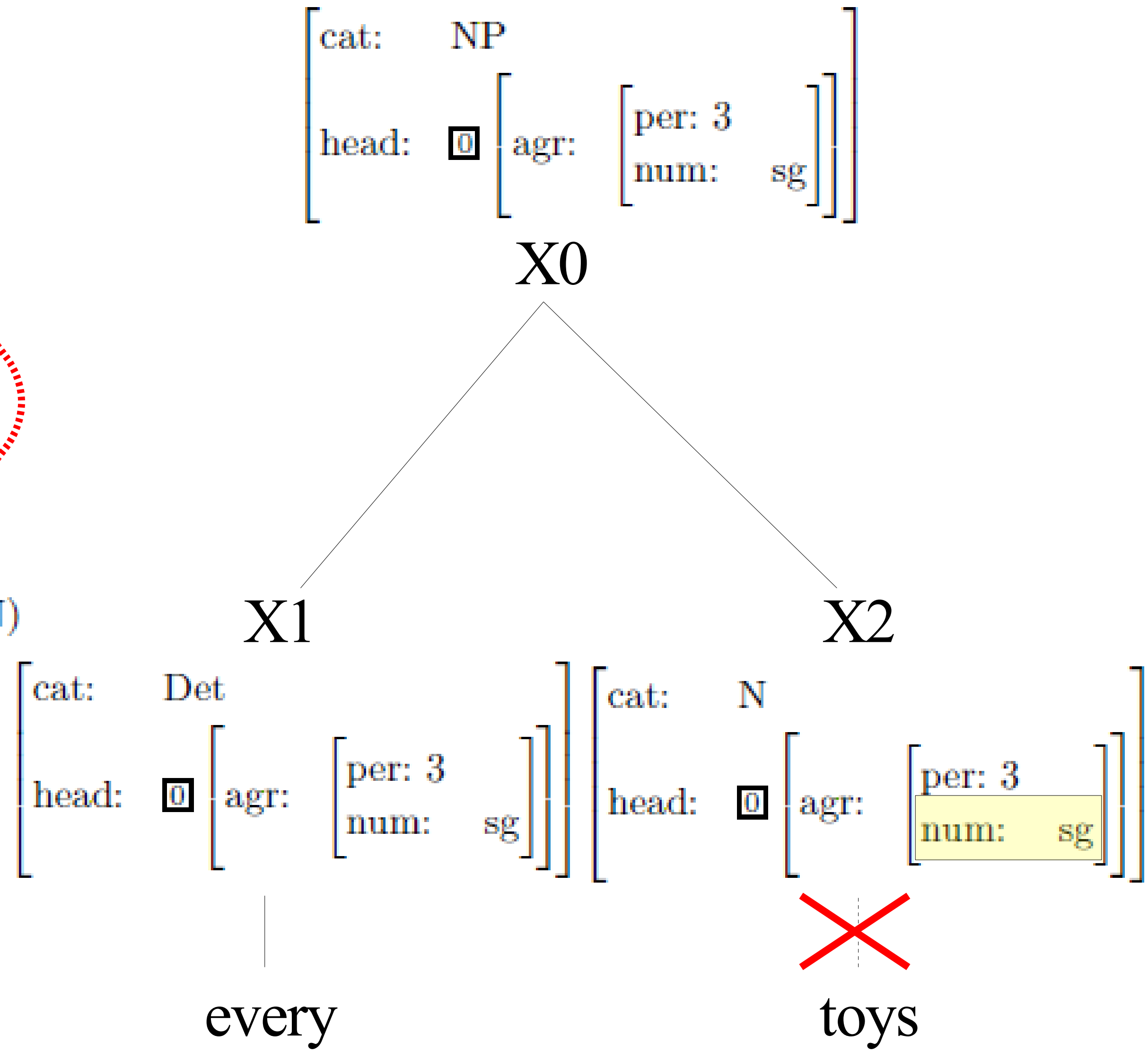
every :=
[cat: Det, head: [agr: [per: 3, num: sg]]]

toys :=
[cat: N, head: [agr: [per: 3, num: pl]]]

- **PS rule:**

X0 → X1 X2 (NP → Det N)

[X0: [cat: NP, head: 0], X1: [cat: Det, head: 0], X2: [cat: N, head: 0]]



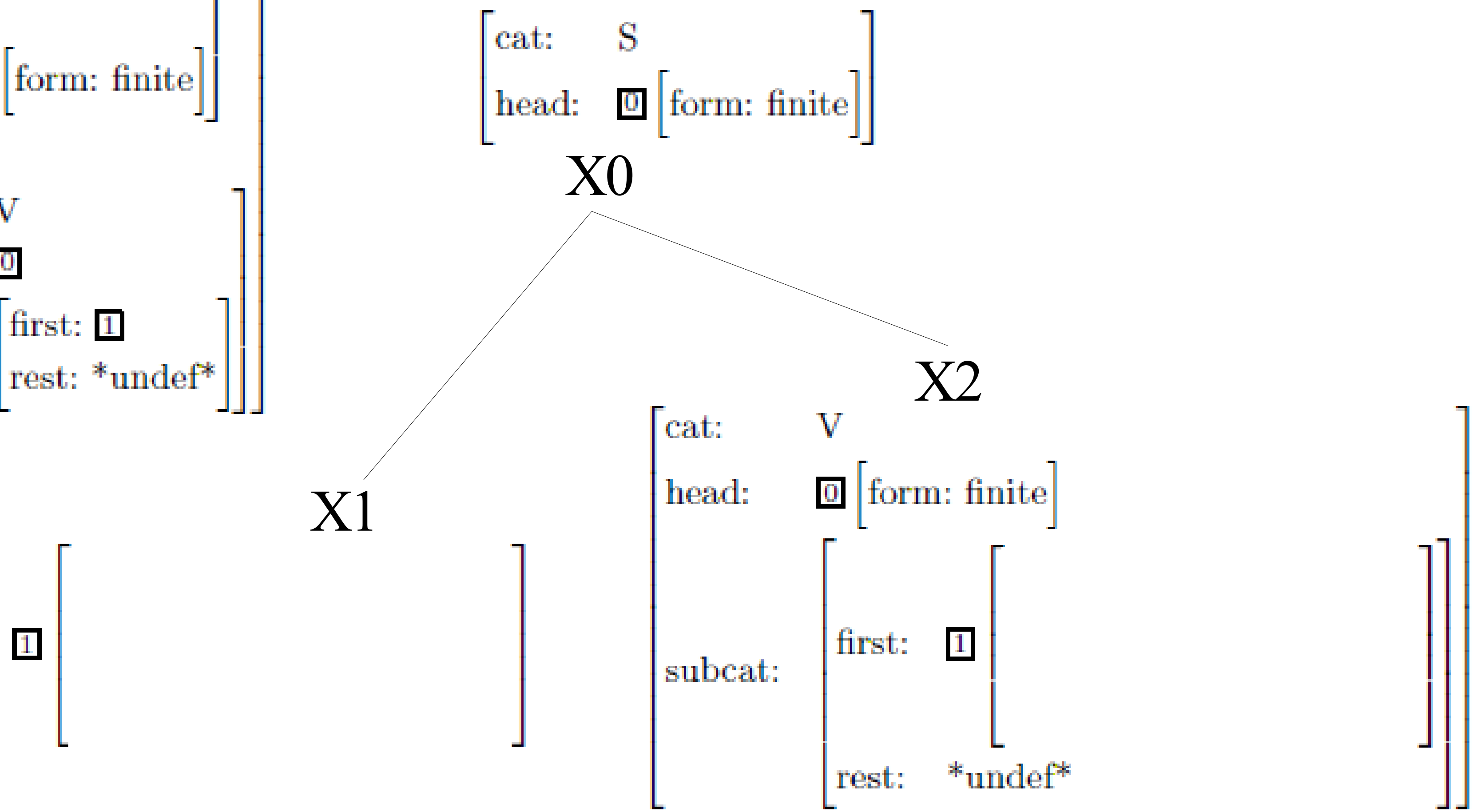
Source: Berthold Crysmann 2006

PATR-II example grammar (*NP snores*)

- **PS rule:**

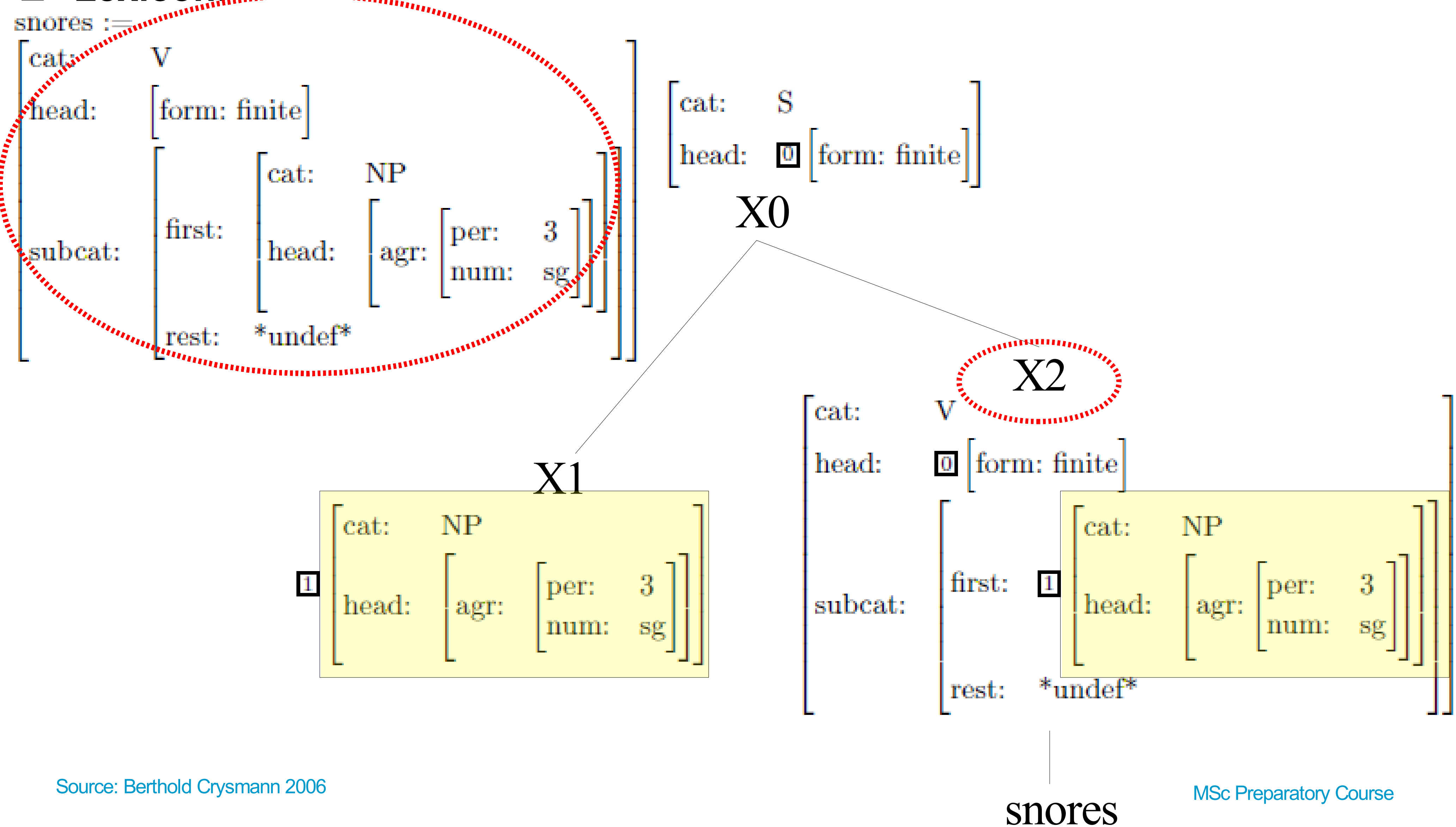
X0 → X1 X2 (S → NP VP)

$$
\begin{bmatrix}
\text{X0:} & \begin{bmatrix} \text{cat:} & \text{S} \\ \text{head:} & \boxed{0}\begin{bmatrix}\text{form: finite}\end{bmatrix} \end{bmatrix} \\
\text{X1:} & \boxed{1} \\
\text{X2:} & \begin{bmatrix} \text{cat:} & \text{V} \\ \text{head:} & \boxed{0} \\ \text{subcat:} & \begin{bmatrix} \text{first:} & \boxed{1} \\ \text{rest:} & \text{*undef*} \end{bmatrix} \end{bmatrix}
\end{bmatrix}
$$



PATR-II example grammar (*NP snores*)

- **Lexicon:**



Source: Berthold Crysmann 2006

PATR-II example grammar (*likes NP*)

- **PS rule:**

$$X0 \rightarrow X1\ X2\ (V \rightarrow V\ XP)$$

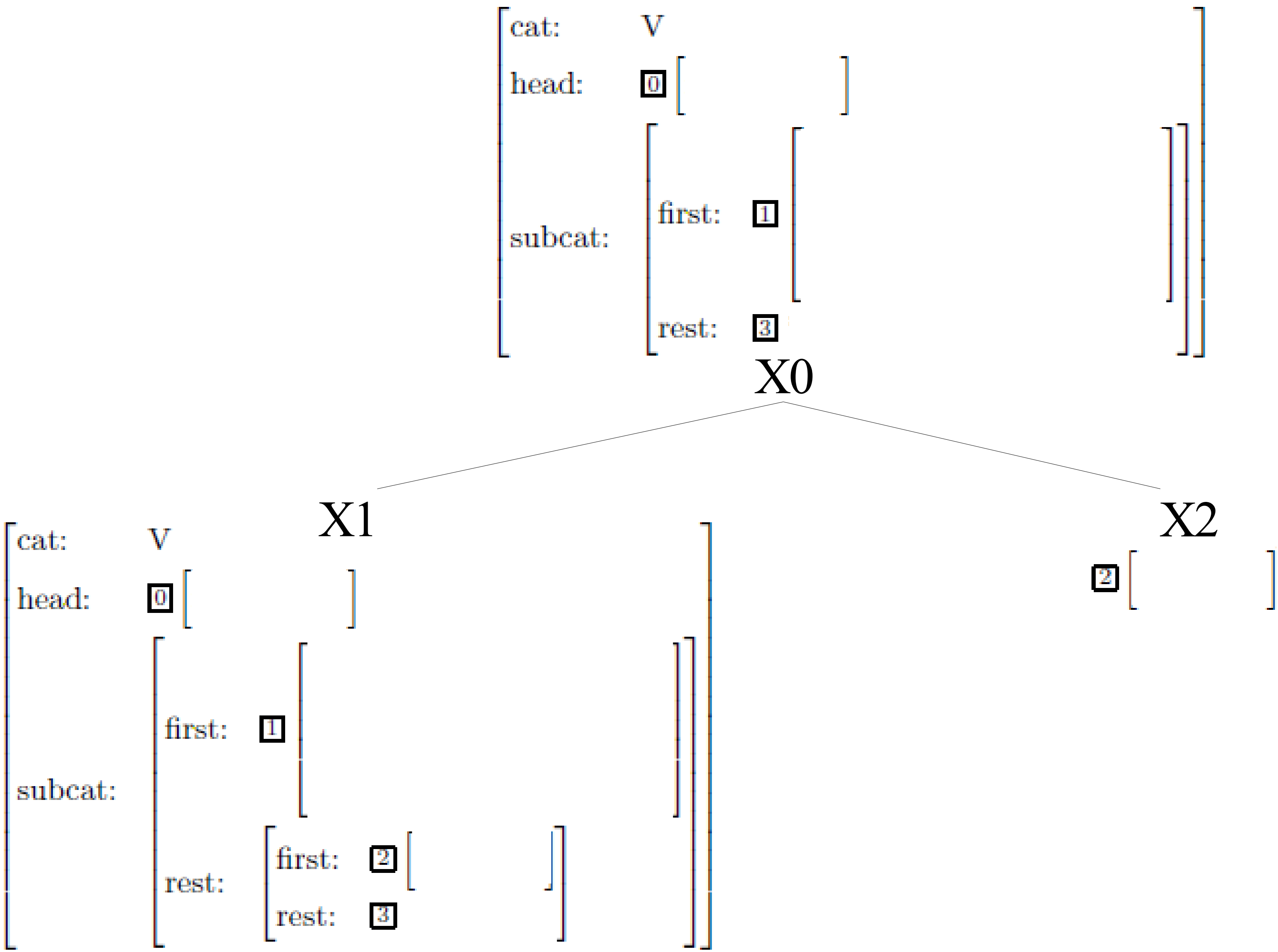
$$\begin{bmatrix} X0: & \begin{bmatrix} \text{cat:} & V \\ \text{head:} & \boxed{0} \\ \text{subcat:} & \begin{bmatrix} \text{first:} & \boxed{1} \\ \text{rest:} & \boxed{3} \end{bmatrix} \end{bmatrix} \\ X1: & \begin{bmatrix} \text{cat:} & V \\ \text{head:} & \boxed{0} \\ \text{subcat:} & \begin{bmatrix} \text{first:} & \boxed{1} \\ \text{rest:} & \begin{bmatrix} \text{first:} & \boxed{2} \\ \text{rest:} & \boxed{3} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ X2: & \boxed{2} \end{bmatrix}$$

- **Lexicon:**

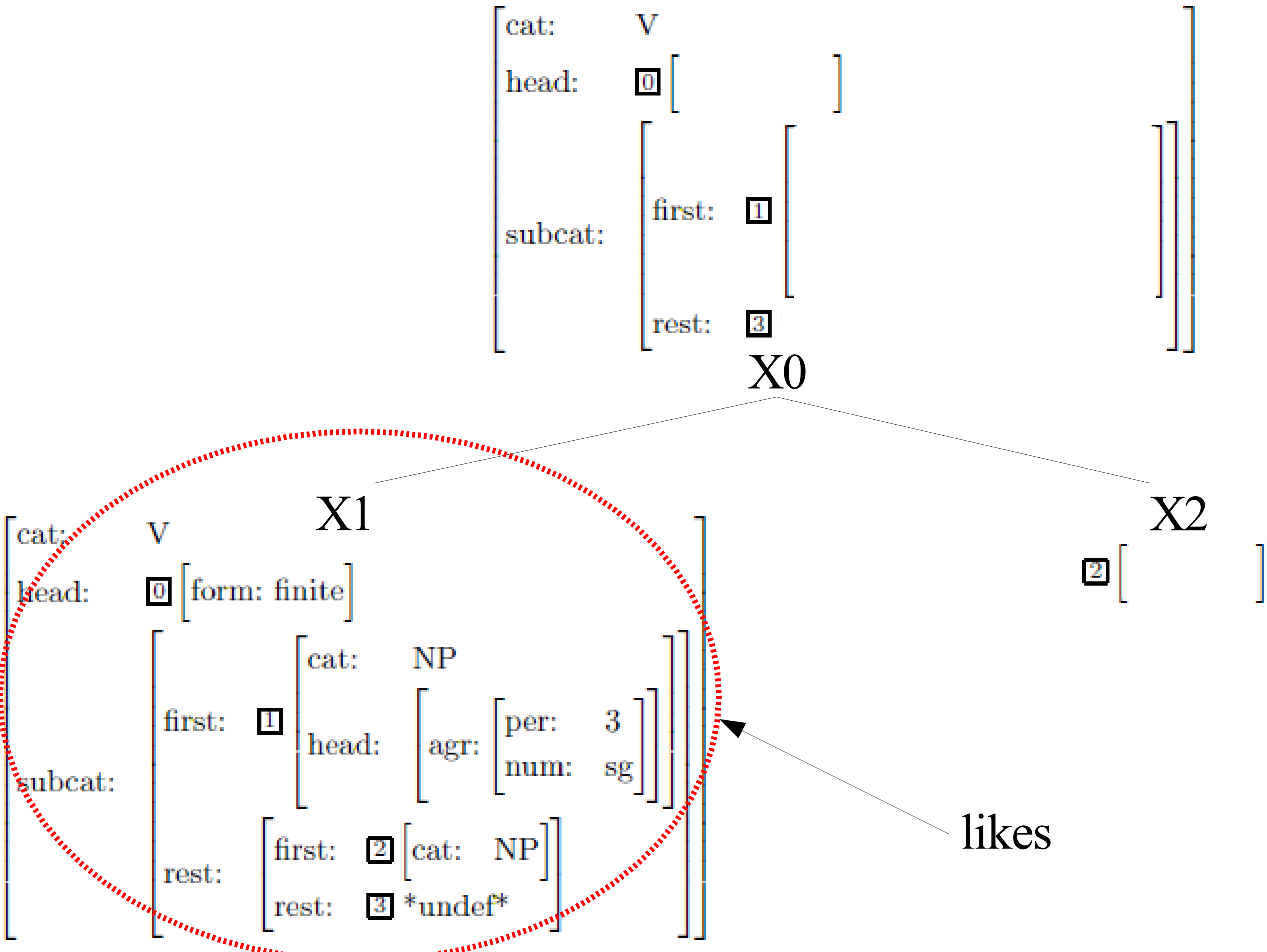
likes :=

$$\begin{bmatrix} \text{cat:} & V \\ \text{head:} & \begin{bmatrix} \text{form:} & \text{finite} \end{bmatrix} \\ \text{subcat:} & \begin{bmatrix} \text{first:} & \begin{bmatrix} \text{cat:} & NP \\ \text{head:} & \begin{bmatrix} \text{agr:} & \begin{bmatrix} \text{per:} & 3 \\ \text{num:} & \text{sg} \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \text{rest:} & \begin{bmatrix} \text{first:} & \begin{bmatrix} \text{cat:} & NP \end{bmatrix} \\ \text{rest:} & \text{*undef*} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

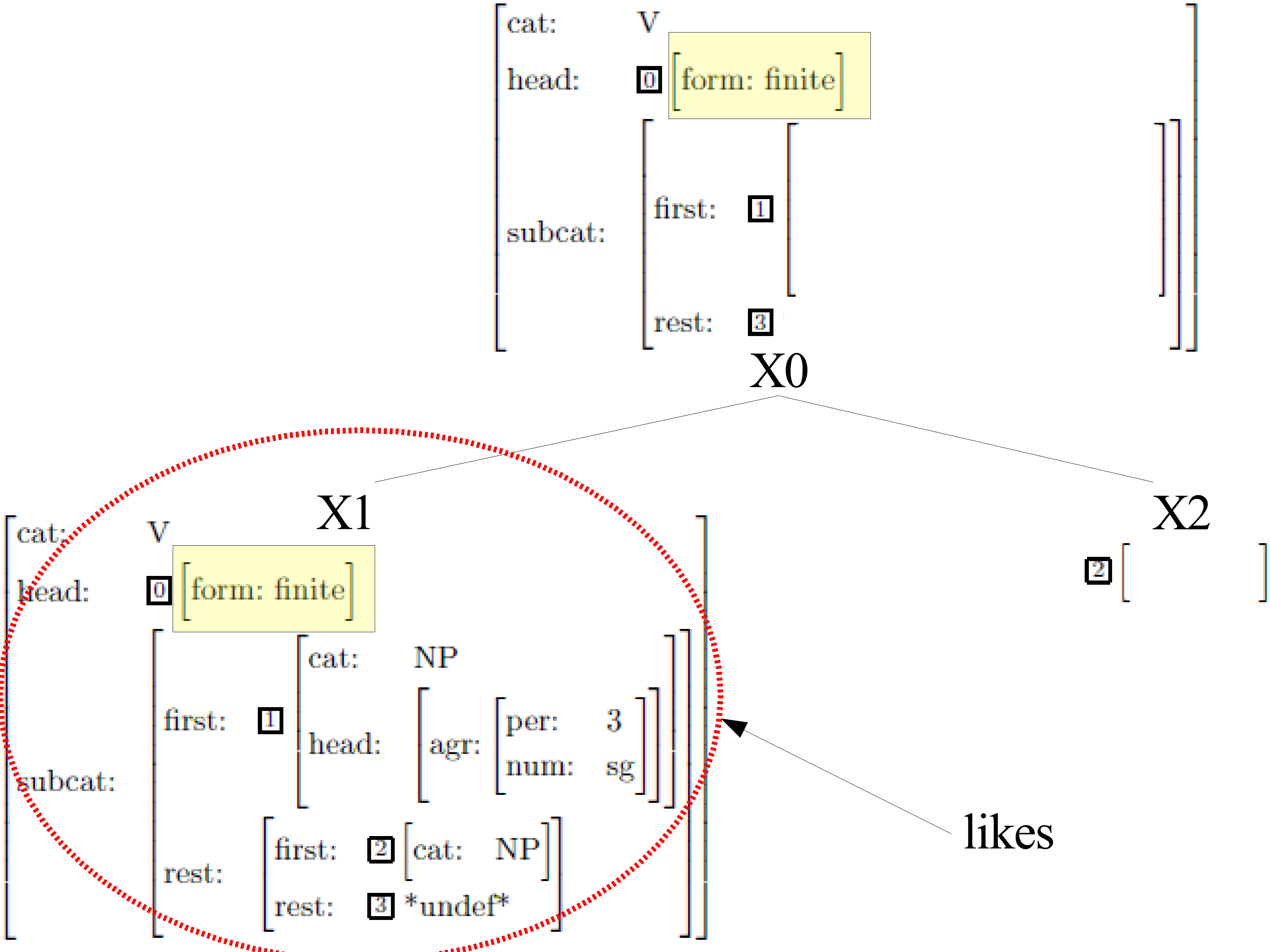
PATR-II example grammar (*likes NP*)



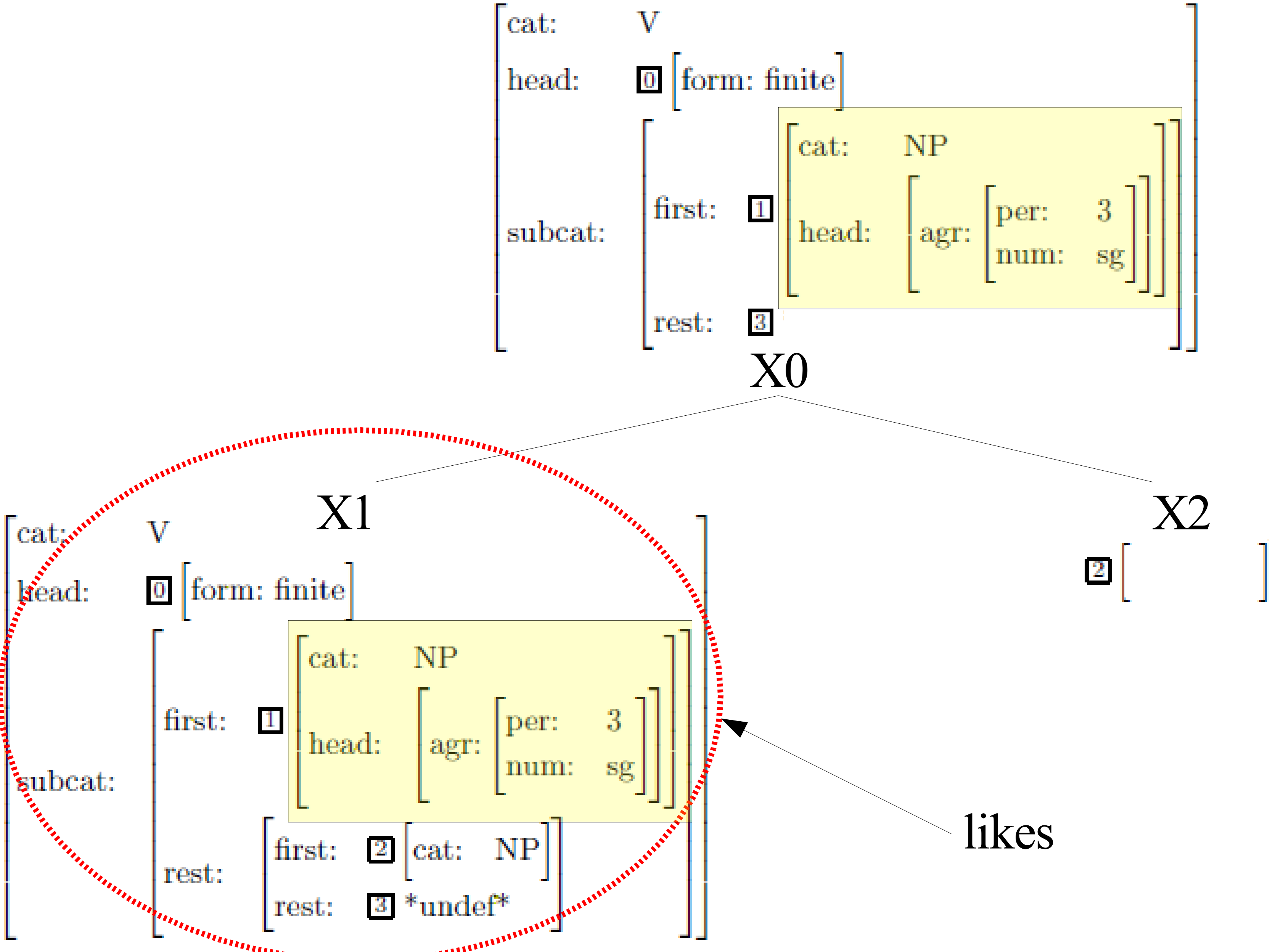
PATR-II example grammar (*likes NP*)



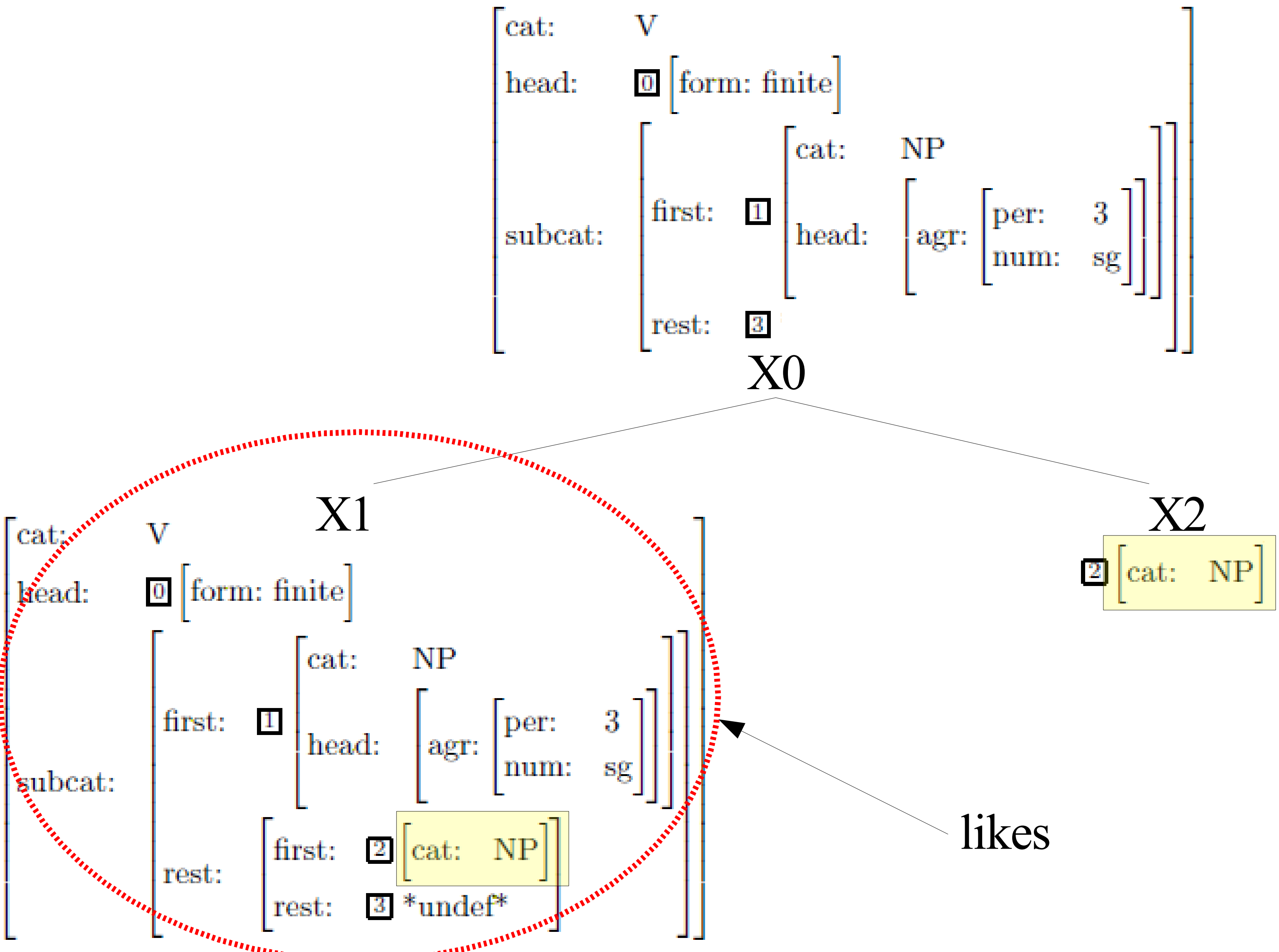
PATR-II example grammar (*likes NP*)



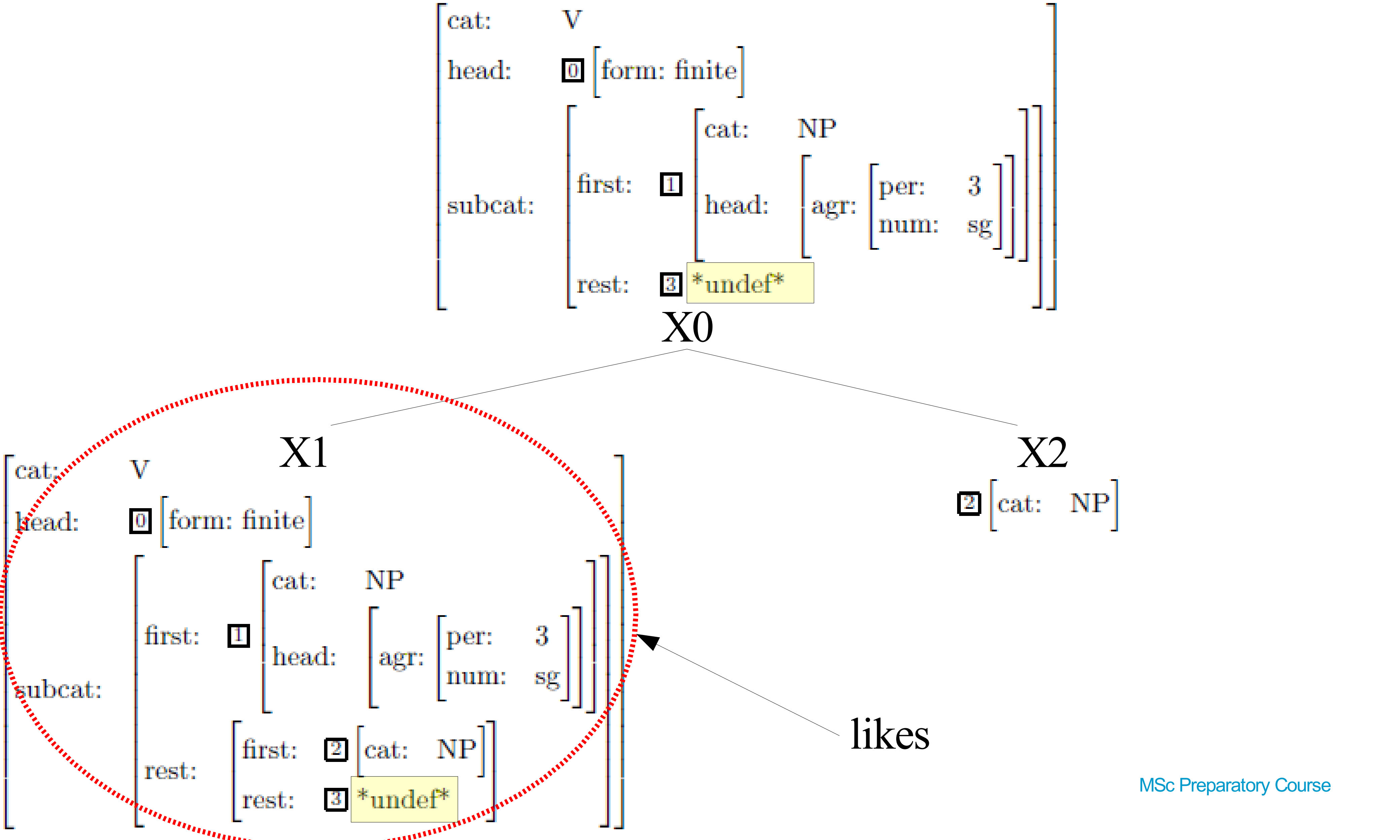
PATR-II example grammar (*likes NP*)



PATR-II example grammar (*likes NP*)



PATR-II example grammar (*likes NP*)



Lexical-Functional Grammar (LFG)

- **Developed by Joan Bresnan & Ron Kaplan (late 70s to early 80s)**
- **Reference:**
  - Bresnan (ed.) 1982, "The mental representation of grammatical relations"
- **Architecture:**
  - Separation of *c*(onstituent)-structure and *f*(unctional)-structure
  - *c*-structure is a context-free phrase structure tree (with functional annotations)
  - *f*-structure is a feature structure encoding grammatical functions
  - Functional annotations constrain the mapping from c-structure nodes to f-structure representations

Lexical-Functional Grammar (LFG)

- ***f*-structure**
  - Attribute value matrix (AVM)
  - Values can be atomic, complex (FS), or sets
  - PRED values are special atomic values
    - Defining the interface for semantic interpretation
    - Encode grammatical functions governed by a predicate
- **Examples:**

$$\begin{bmatrix} \text{SUBJ} & \begin{bmatrix} \text{CASE} & \text{`NOM'} \\ \text{NUM} & \text{`SG'} \\ \text{PER} & \text{`3'} \end{bmatrix} \end{bmatrix} \quad \begin{bmatrix} \text{PRED} & \text{`snore}\langle(\uparrow\text{SUBJ})\rangle\text{'} \end{bmatrix}$$

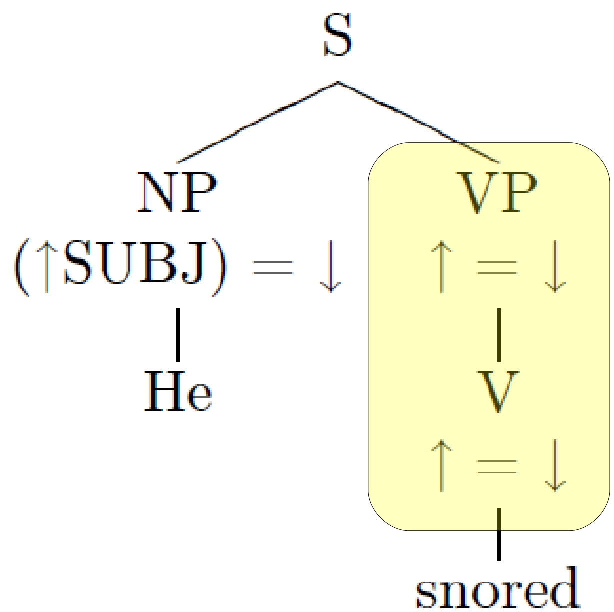
Source: Berthold Crysmann 2006

Lexical-Functional Grammar (LFG)

- ***c*-structure**
  - Licensed by context-free phrase structure rules
  - PS-rules augmented with optionality, disjunction, Kleene *
  - Functional annotations define mapping into *f*-structure
- **Examples:**

$$S \rightarrow \underset{(\uparrow \text{SUBJ}) = \downarrow}{\text{NP}} \quad \underset{\uparrow = \downarrow}{\text{VP}}$$

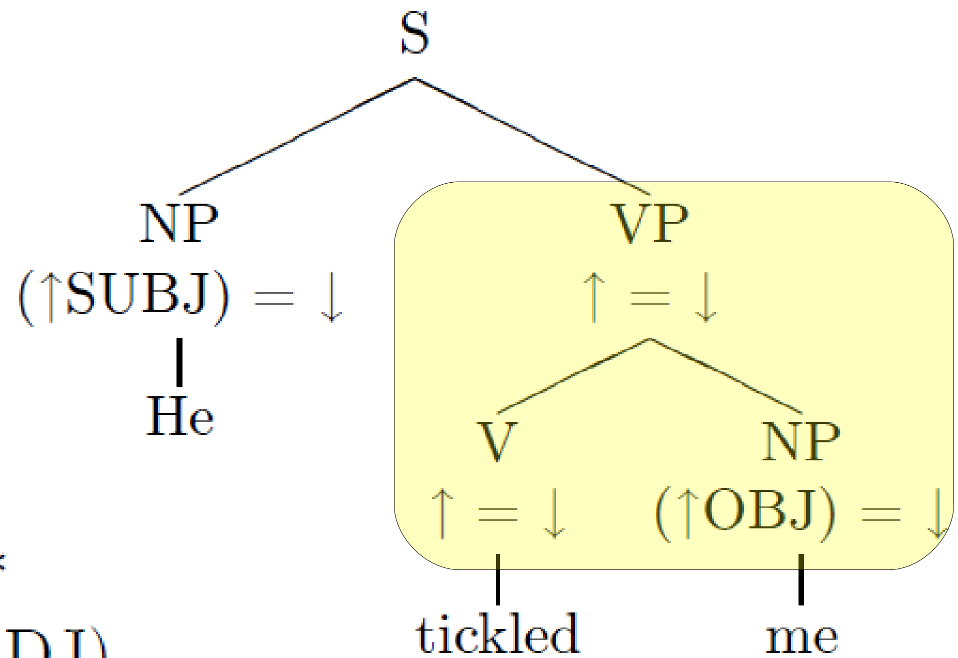
$$\text{VP} \rightarrow \underset{\uparrow = \downarrow}{\text{V}} \quad \underset{(\uparrow \text{OBJ}) = \downarrow}{(\text{NP})} \quad \underset{\downarrow \in (\uparrow \text{ADJ})}{\text{PP*}}$$



Lexical-Functional Grammar (LFG)

- ***c*-structure**
  - Licensed by context-free phrase structure rules
  - PS-rules augmented with optionality, disjunction, Kleene *
  - Functional annotations define mapping into *f*-structure
- **Examples:**

$$
\begin{array}{llll}
S \rightarrow & NP & VP & \\
 & (\uparrow SUBJ) = \downarrow & \uparrow = \downarrow & \\
VP \rightarrow & V & (NP) & PP^{*} \\
 & \uparrow = \downarrow & (\uparrow OBJ) = \downarrow & \downarrow \in (\uparrow ADJ)
\end{array}
$$

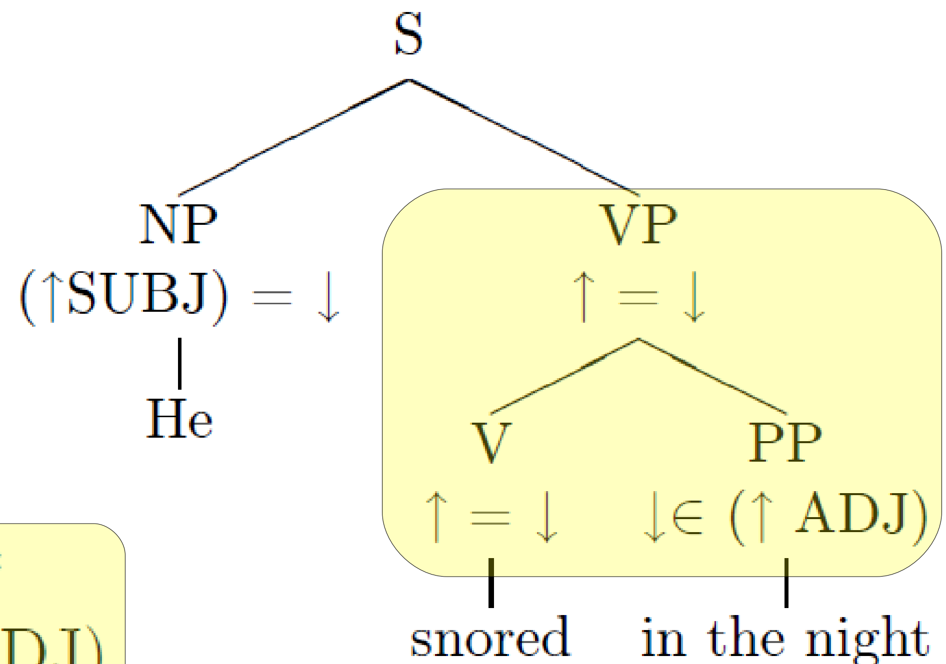


Lexical-Functional Grammar (LFG)

- ***c*-structure**
  - Licensed by context-free phrase structure rules
  - PS-rules augmented with optionality, disjunction, Kleene *
  - Functional annotations define mapping into *f*-structure
- **Examples:**

$$S \rightarrow \underset{(\uparrow \text{SUBJ}) = \downarrow}{\text{NP}} \quad \underset{\uparrow = \downarrow}{\text{VP}}$$

$$\text{VP} \rightarrow \underset{\uparrow = \downarrow}{\text{V}} \quad \underset{(\uparrow \text{OBJ}) = \downarrow}{(\text{NP})} \quad \underset{\downarrow \in (\uparrow \text{ADJ})}{\text{PP*}}$$



Lexical-Functional Grammar (LFG)

- **Lexicon**
  - Lexical entries are constraints over (terminal) *c*-structure nodes and their associated *f*-structure representations
- **Examples:**
  - “he” NP (↑PRED) = `PRO'
    (↑CASE) = `NOM'
    (↑NUM) = `SG'
    (↑PER) = `3'

$$\begin{bmatrix} \text{PRED} & \text{‘PRO’} \\ \text{CASE} & \text{‘NOM’} \\ \text{NUM} & \text{‘SG’} \\ \text{PER} & \text{‘3’} \end{bmatrix}$$

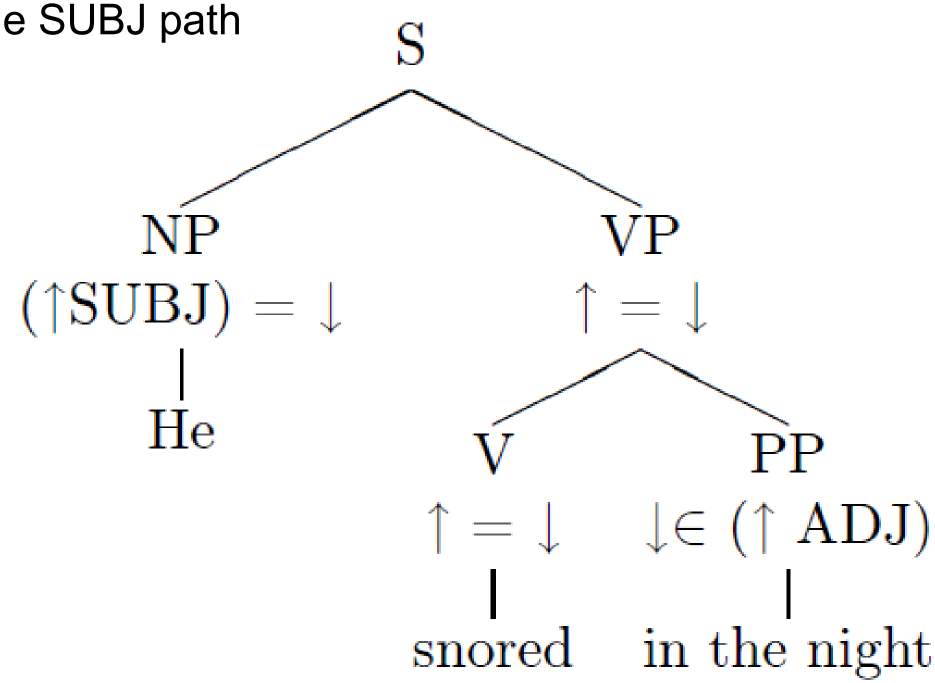
  - “snored” V (↑PRED) = `snored<(↑SUBJ)>'

$$\begin{bmatrix} \text{PRED} & \text{‘snore}\langle\left(\uparrow\text{SUBJ}\right)\rangle\text{’} \end{bmatrix}$$

Source: Berthold Crysmann 2006

Lexical-Functional Grammar (LFG)

- **Mapping from *c*-structure to *f*-structure**
  - Functional designator ↑ refers to *f*-structure associated with mother node
  - Functional designator ↓ refers to a node's own *f*-structure
- **Examples:**
  - ↑ = ↓
    - Identifies a node's *f*-structure with that of its mother
  - (↑SUBJ) = ↓
    - Identifies a node's *f*-structure with the SUBJ path of it's mother's *f*-structure

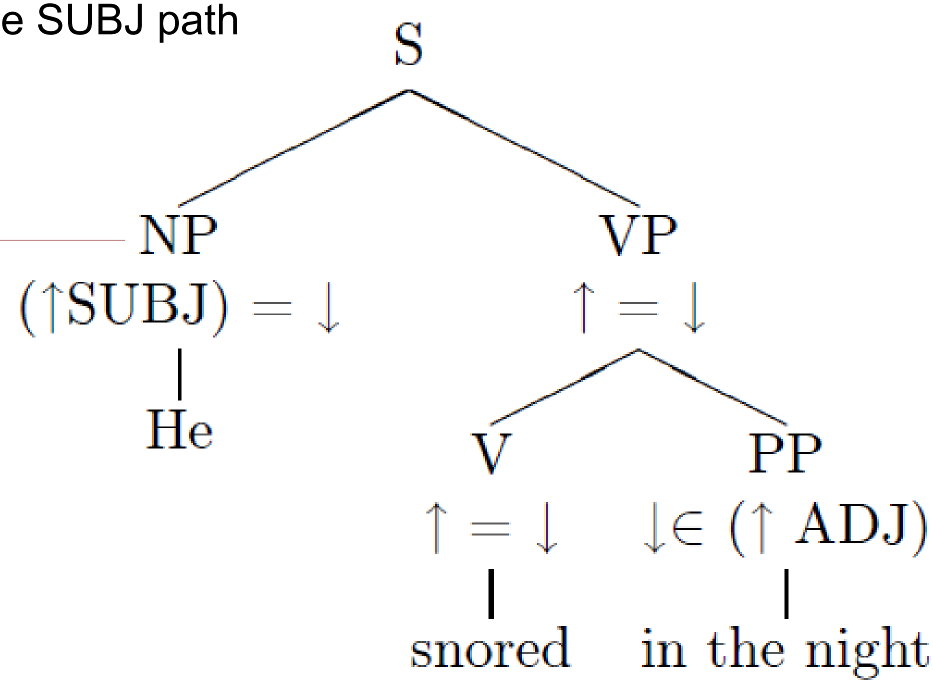


Source: Berthold Crysmann 2006

Lexical-Functional Grammar (LFG)

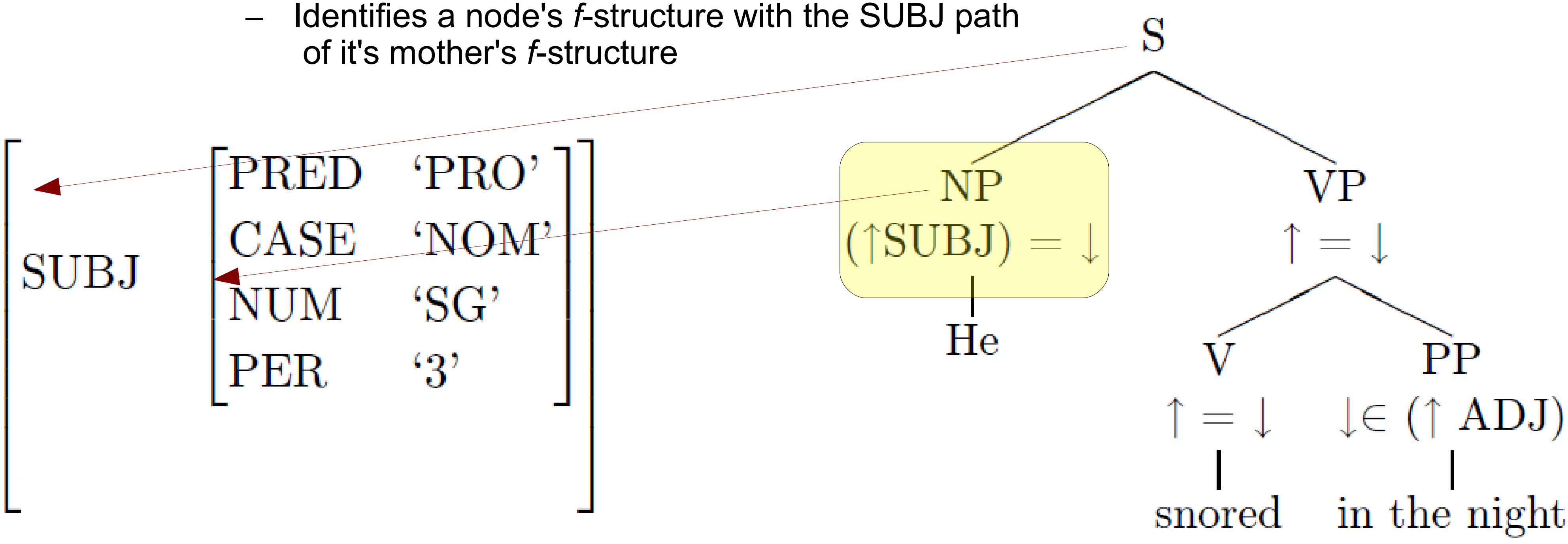
- **Mapping from *c*-structure to *f*-structure**
  - Functional designator ↑ refers to *f*-structure associated with mother node
  - Functional designator ↓ refers to a node's own *f*-structure
- **Examples:**
  - ↑ = ↓
    - Identifies a node's *f*-structure with that of its mother
  - (↑SUBJ) = ↓
    - Identifies a node's *f*-structure with the SUBJ path of it's mother's *f*-structure

$$\begin{bmatrix} \text{PRED} & \text{`PRO'} \\ \text{CASE} & \text{`NOM'} \\ \text{NUM} & \text{`SG'} \\ \text{PER} & \text{`3'} \end{bmatrix}$$



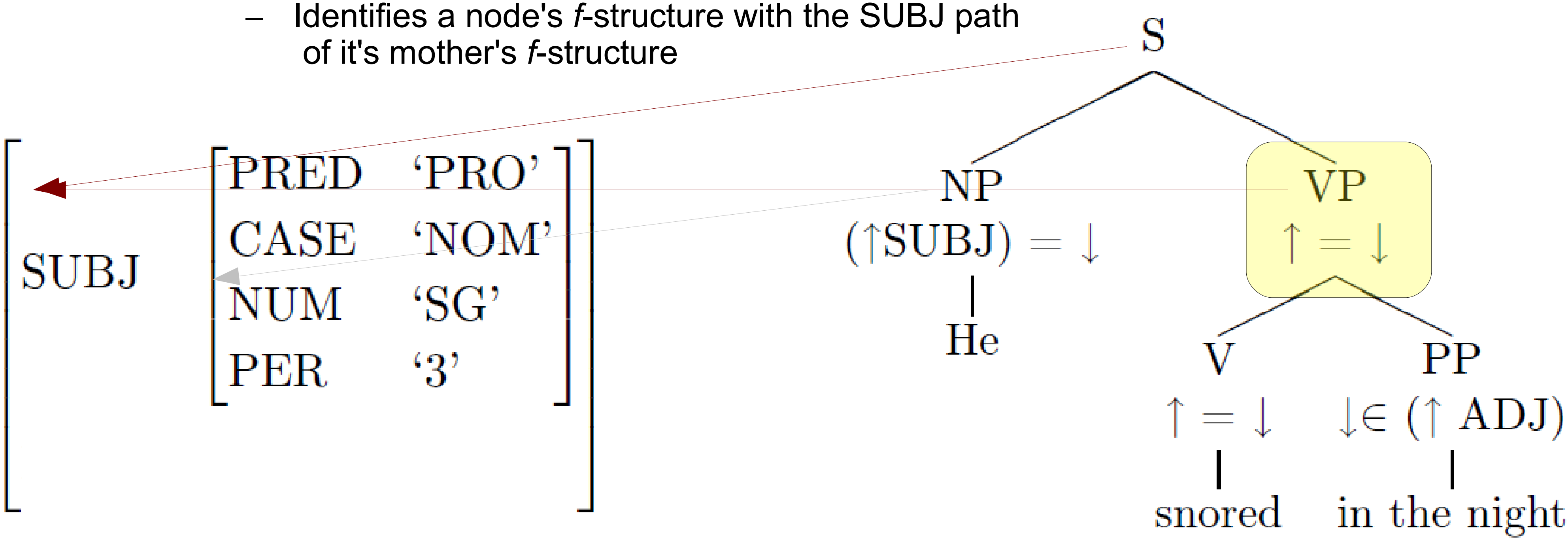
Lexical-Functional Grammar (LFG)

- **Mapping from *c*-structure to *f*-structure**
  - Functional designator ↑ refers to *f*-structure associated with mother node
  - Functional designator ↓ refers to a node's own *f*-structure
- **Examples:**
  - ↑ = ↓
    - Identifies a node's *f*-structure with that of its mother
  - (↑SUBJ) = ↓
    - Identifies a node's *f*-structure with the SUBJ path of it's mother's *f*-structure



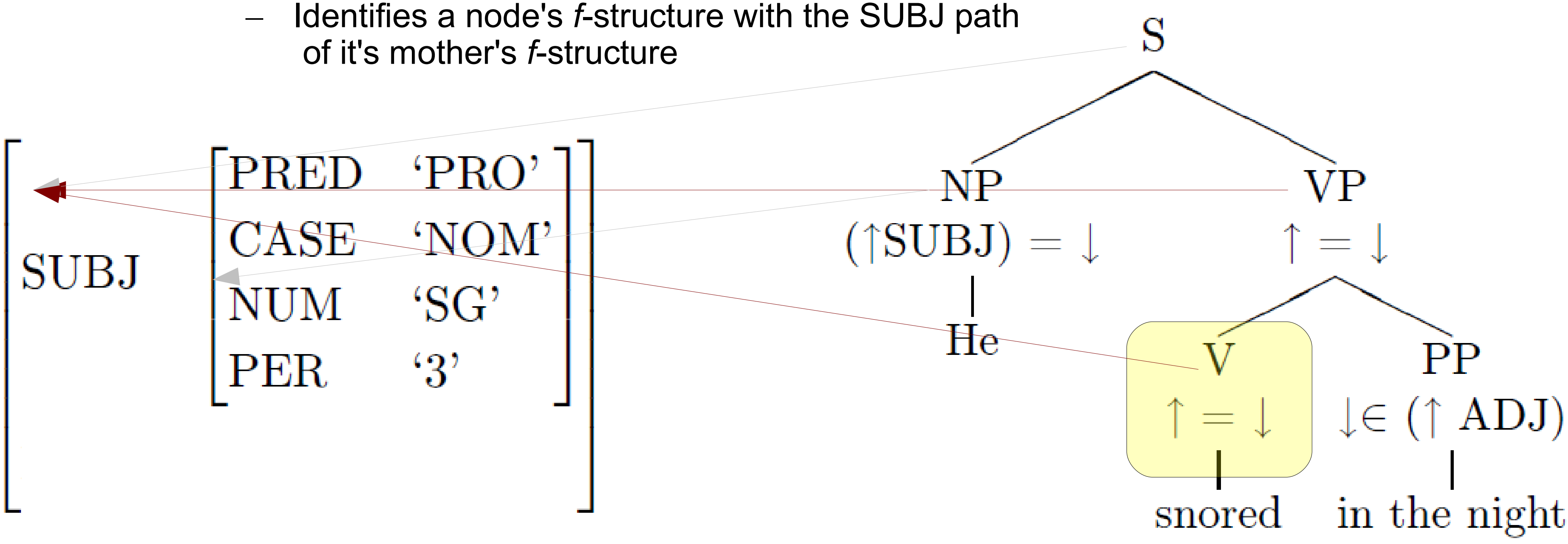
Lexical-Functional Grammar (LFG)

- **Mapping from *c*-structure to *f*-structure**
  - Functional designator ↑ refers to *f*-structure associated with mother node
  - Functional designator ↓ refers to a node's own *f*-structure
- **Examples:**
  - ↑ = ↓
    - Identifies a node's *f*-structure with that of its mother
  - (↑SUBJ) = ↓
    - Identifies a node's *f*-structure with the SUBJ path of it's mother's *f*-structure



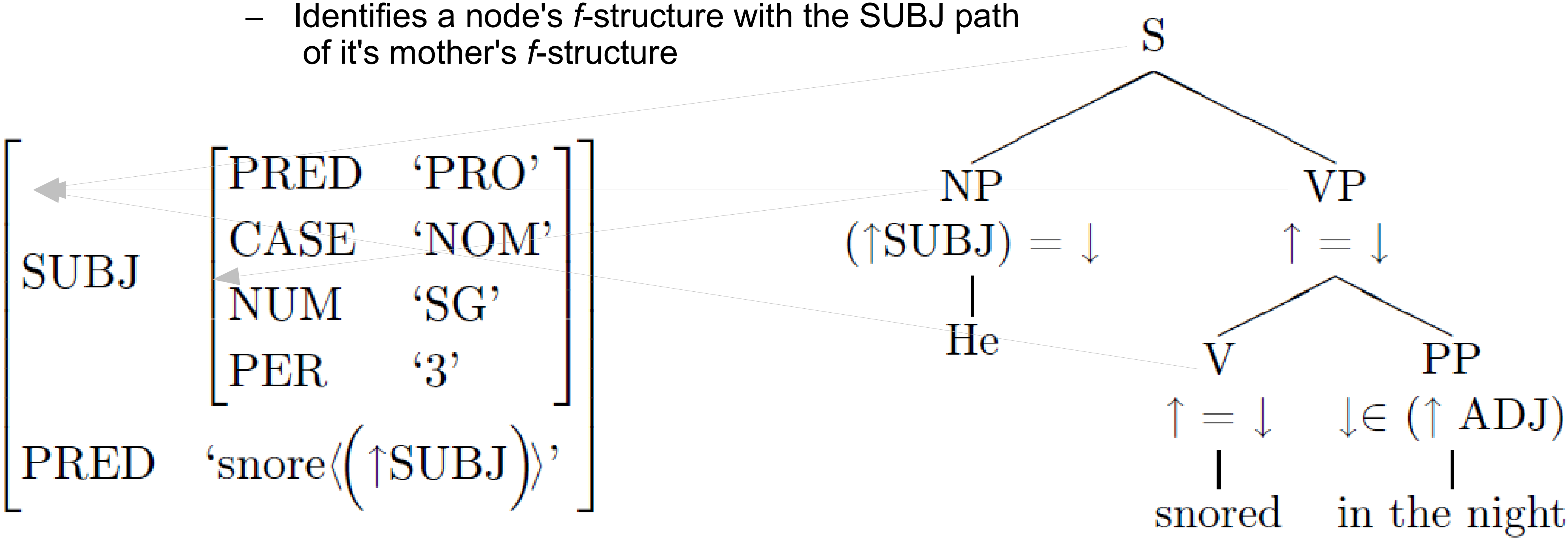
Lexical-Functional Grammar (LFG)

- **Mapping from *c*-structure to *f*-structure**
  - Functional designator ↑ refers to *f*-structure associated with mother node
  - Functional designator ↓ refers to a node's own *f*-structure
- **Examples:**
  - ↑ = ↓
    - Identifies a node's *f*-structure with that of its mother
  - (↑SUBJ) = ↓
    - Identifies a node's *f*-structure with the SUBJ path of it's mother's *f*-structure



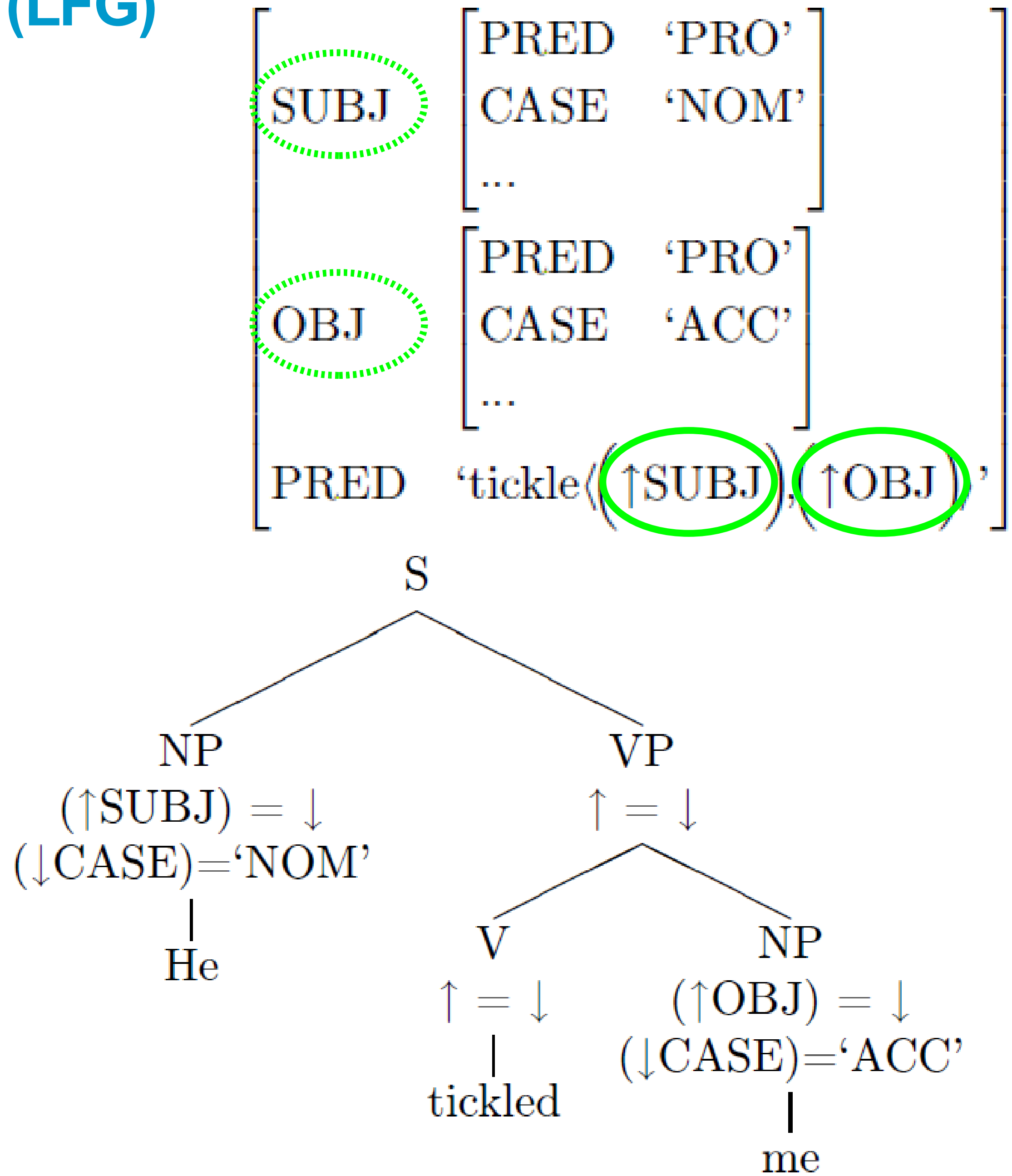
Lexical-Functional Grammar (LFG)

- **Mapping from *c*-structure to *f*-structure**
  - Functional designator ↑ refers to *f*-structure associated with mother node
  - Functional designator ↓ refers to a node's own *f*-structure
- **Examples:**
  - ↑ = ↓
    - Identifies a node's *f*-structure with that of its mother
  - (↑SUBJ) = ↓
    - Identifies a node's *f*-structure with the SUBJ path of it's mother's *f*-structure



Lexical-Functional Grammar (LFG)

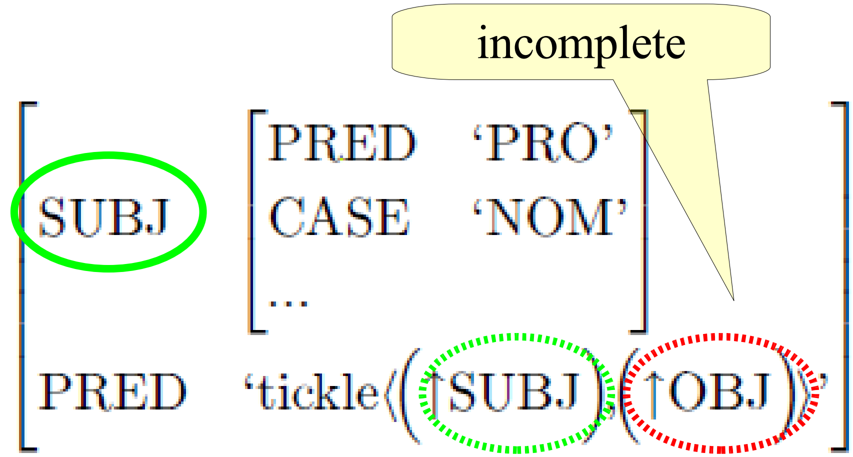
- ***f*-structure wellformedness conditions**
  - Functional Uniqueness
  - Functional Completeness
    "An f-structure is locally complete [iff] it contains all the governable functions that its predicate governs."
    (Kaplan & Bresnan 1982)
  - Functional Coherence
    "An f-structure is locally coherent [iff] all the governable functions that it contains are governed by a local predicate."
    (Kaplan & Bresnan 1982)

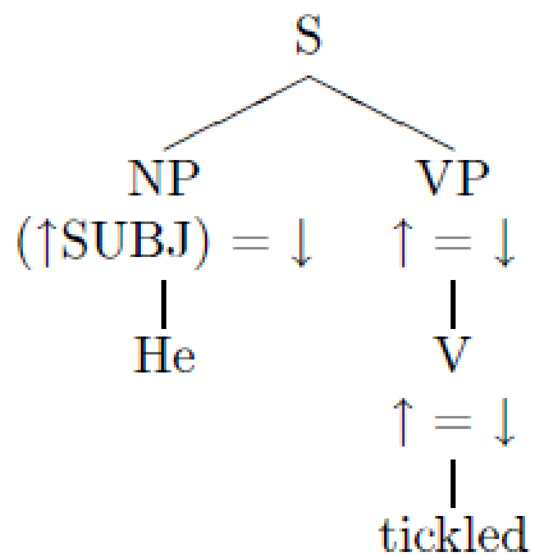


Source: Berthold Crysmann 2006

Lexical-Functional Grammar (LFG)

- ***f*-structure wellformedness conditions**
  - Functional Uniqueness
  - Functional Completeness
    "An f-structure is locally complete [iff] it contains all the governable functions that its predicate governs."
    (Kaplan & Bresnan 1982)
  - Functional Coherence
    "An f-structure is locally coherent [iff] all the governable functions that it contains are governed by a local predicate."
    (Kaplan & Bresnan 1982)

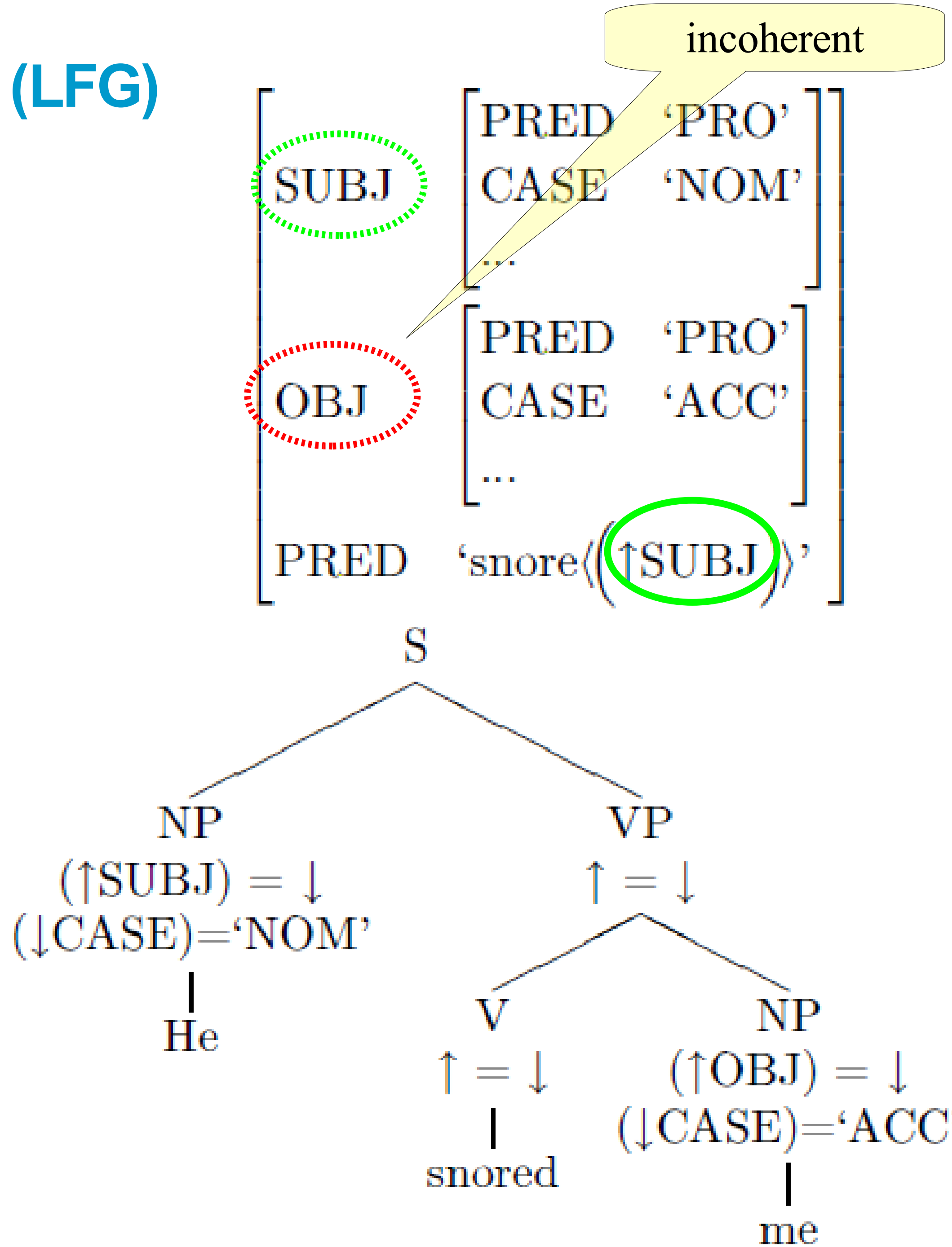




Source: Berthold Crysmann 2006

Lexical-Functional Grammar (LFG)

- ***f*-structure wellformedness conditions**
  - Functional Uniqueness
  - Functional Completeness
    "An f-structure is locally complete [iff] it contains all the governable functions that its predicate governs."
    (Kaplan & Bresnan 1982)
  - Functional Coherence
    "An f-structure is locally coherent [iff] all the governable functions that it contains are governed by a local predicate."
    (Kaplan & Bresnan 1982)



Source: Berthold Crysmann 2006

Grammar formalism and grammatical theory

- **Grammar formalisms**
  - Set of data structures (trees/DAGs) and operations (substitution, unification) that permit to
    - Describe the set of strings of a language
    - Capture grammatical relations
- **Grammatical theory**
  - Universal inventory of descriptive devices make predictions about language
  - Study of individual languages contributes to a theory of language
- **Examples**
  - LFG (grammar formalism with an accompanying theory)
    - LFG's architecture is a hypothesis about the structure of languages (modules)
    - Finite set of governable grammatical functions (SUBJ,OBJ,COMP,XCOMP) is assumed to be applicable to all languages (universal)
  - PATR- II (pure formalism)
    - Inventory of features unconstrained
    - Different theories implementable with PATR-formalism, e.g., PSG, or Categorial Unification Grammar (Uszkoreit, 1987)

Source: Berthold Crysmann 2006

Outlook

- **Unification-based grammars**
  - The tool of choice for developing high-precision grammars
  - Indispensible for manual grammar development
- **Current UBGs grounded in syntactic theories**
  - LFG
  - HPSG (Pollard & Sag 1987,1994)
- **UBGs can be processed efficiently**
  - LFG: XLE (Xerox)
  - HPSG: LKB (Copestake 2001) & Pet (Callmeier, 2000)
- **Compilation (approximation) into leaner formalisms possible**
  - TAG
  - CFG

Source: Berthold Crysmann 2006

References

- **General:**
  - Shieber, S. M. (1986) An Introduction to Unification-Based Approaches to Grammar. CSLI Lecture Notes 4. CSLI, Stanford, CA.
- **LFG:**
  - Bresnan, J. (Ed.) (1982) The Mental Representation of Grammatical Relations. MIT Press, Cambridge, Mass.
  - Dalrymple, Mary, Ronald M. Kaplan, John T. Maxwell, and Annie Zaenen, editors. 1995a. Formal Issues in Lexical-Functional Grammar. CSLI Publications, Stanford, CA.
- **HPSG:**
  - Pollard, C. and I. A. Sag (1994) Head-Driven Phrase Structure Grammar. Chicago: University of Chicago Press.
  - Copestake (2001) Implementing typed feature structures, CSLI publications, Stanford.