Feature-based Parsing

Deep Processing for NLP
Ling 571
February 5, 2014

Roadmap

- Features: Motivation
  - Constraint & compactness
- Features
  - Definitions & representations
- Unification
- Application of features in the grammar
  - Agreement, subcategorization
- Parsing with features & unification
  - Augmenting the Earley parser, unification parsing
- Extensions: Types, inheritance, etc
- Conclusion

Constraints & Compactness

- Constraints in grammar
  - S → NP VP
    - They run.
    - He runs.

Constraints & Compactness

- Constraints in grammar
  - S → NP VP
    - They run.
    - He runs.
  - But…
    - *They runs
    - *He run
    - *He disappeared the flight

Constraints & Compactness

- Constraints in grammar
  - S → NP VP
    - They run.
    - He runs.
  - But…
    - *They runs
    - *He run
    - *He disappeared the flight
  - Violate agreement (number), subcategorization

Enforcing Constraints

- Enforcing constraints

Enforcing Constraints

- Enforcing constraints
  - Add categories, rules

Enforcing Constraints

- Enforcing constraints
  - Add categories, rules
    - Agreement:
      - S→ NPsg3p VPsg3p,
      - S→ NPpl3p VPpl3p,

Enforcing Constraints

- Enforcing constraints
  - Add categories, rules
    - Agreement:
      - S→ NPsg3p VPsg3p,
      - S→ NPpl3p VPpl3p,
    - Subcategorization:
      - VP → Vtrans NP,
      - VP → Vintrans,
      - VP → Vditrans NP NP

Enforcing Constraints

- Enforcing constraints
  - Add categories, rules
    - Agreement:
      - S→ NPsg3p VPsg3p,
      - S→ NPpl3p VPpl3p,
    - Subcategorization:
      - VP → Vtrans NP,
      - VP → Vintrans,
      - VP → Vditrans NP NP
  - Explosive!, loses key generalizations

Why features?

- Need compact, general constraints
  - S → NP VP

Why features?

- Need compact, general constraints
  - S → NP VP
    - Only if NP and VP agree

Why features?

- Need compact, general constraints
  - S → NP VP
    - Only if NP and VP agree
- How can we describe agreement, subcat?

Why features?

- Need compact, general constraints
  - S → NP VP
    - Only if NP and VP agree
- How can we describe agreement, subcat?
  - Decompose into elementary features that must be consistent
  - E.g. Agreement

Why features?

- Need compact, general constraints
  - S → NP VP
    - Only if NP and VP agree
- How can we describe agreement, subcat?
  - Decompose into elementary features that must be consistent
  - E.g. Agreement
    - Number, person, gender, etc

Why features?

- Need compact, general constraints
  - S → NP VP
    - Only if NP and VP agree
- How can we describe agreement, subcat?
  - Decompose into elementary features that must be consistent
  - E.g. Agreement
    - Number, person, gender, etc
- Augment CF rules with feature constraints
  - Develop mechanism to enforce consistency
  - Elegant, compact, rich representation

Feature Representations

- Fundamentally, Attribute-Value pairs
  - Values may be symbols or feature structures
    - Feature path: list of features in structure to value
    - “Reentrant feature structures”: share some struct
  - Represented as
    - Attribute-value matrix (AVM), or
    - Directed acyclic graph (DAG)

AVM

$$\begin{pmatrix} \text{NUMBER} & \text{PL} \end{pmatrix}$$

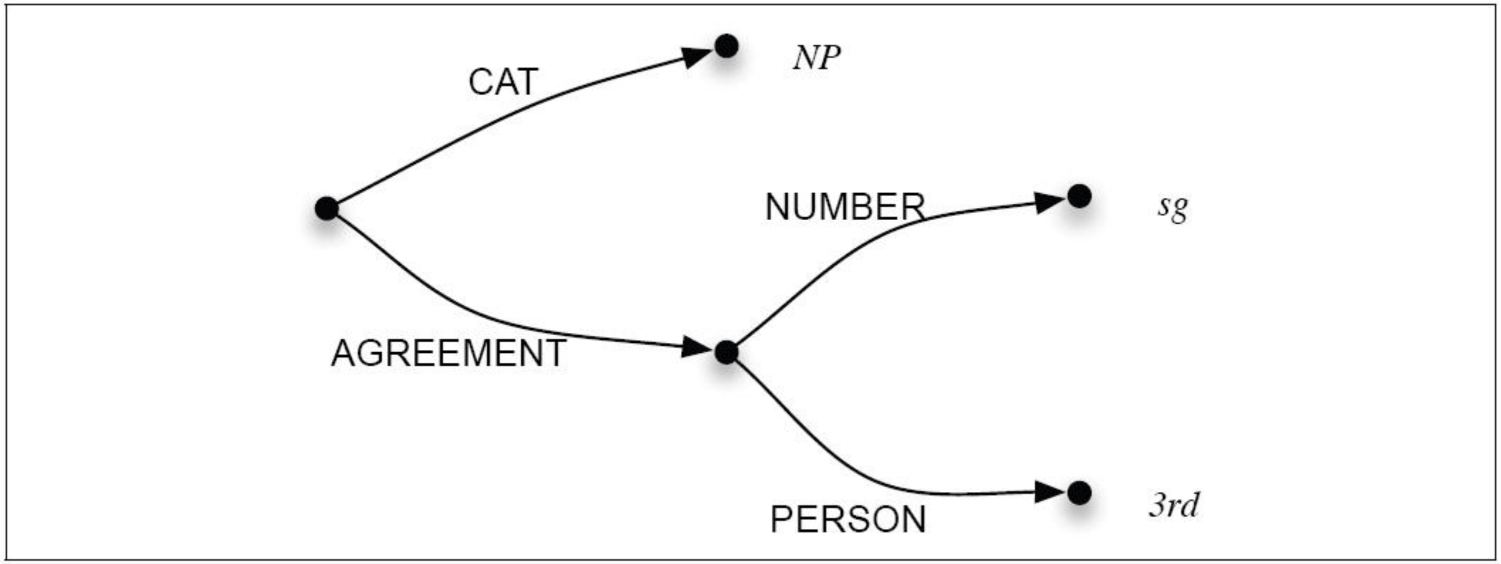
$$\begin{pmatrix} \text{PERSON} & 3 \end{pmatrix}$$

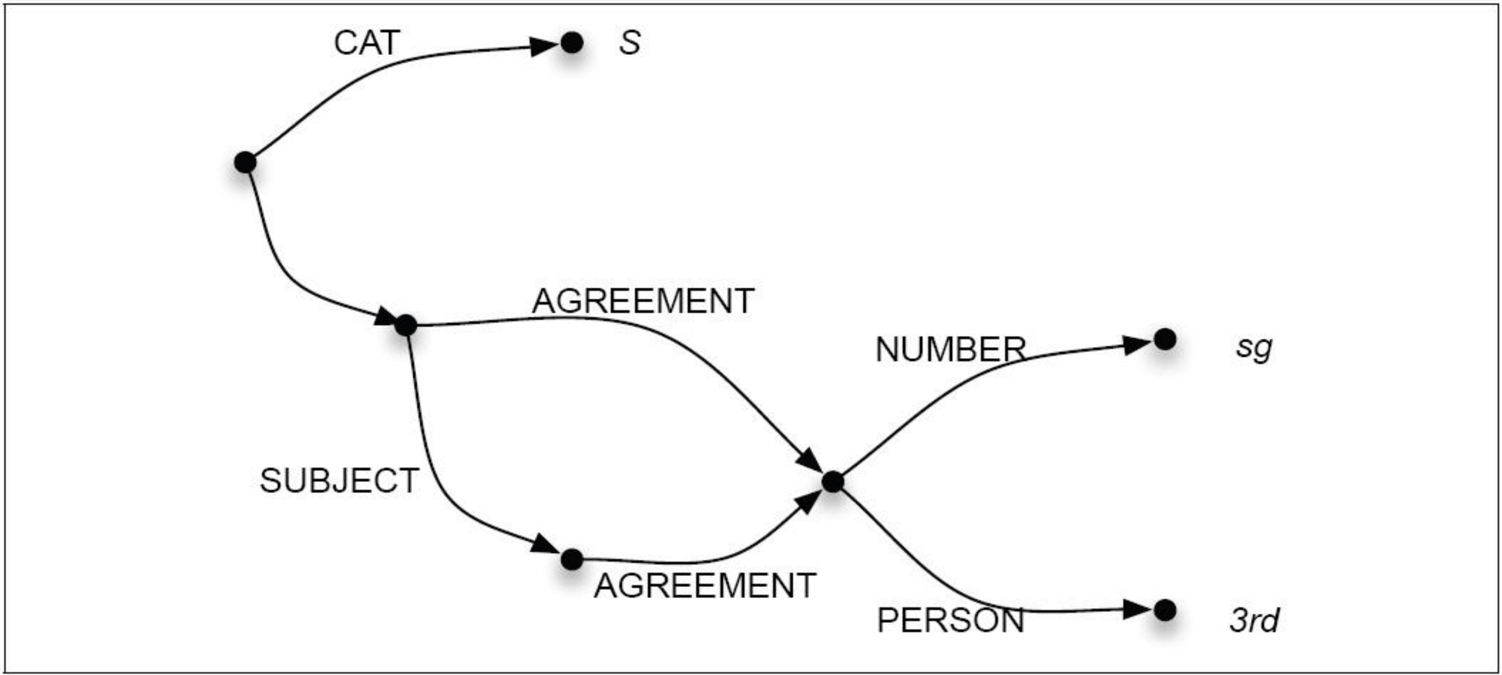
$$\begin{pmatrix} \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{pmatrix}$$

$$\begin{pmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{pmatrix}$$

$$\begin{pmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & \begin{pmatrix} \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{pmatrix} \end{pmatrix}$$

$$\begin{pmatrix} \text{CAT} & \text{S} \\ \text{HEAD} & \begin{pmatrix} \text{AGREEM'T} & \boxed{1} \begin{pmatrix} \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{pmatrix} \\ \text{SUBJECT} & \begin{pmatrix} \text{AGREEMENT} & \boxed{1} \end{pmatrix} \end{pmatrix} \end{pmatrix}$$


CAT
NP
NUMBER
sg
AGREEMENT
PERSON
3rd


CAT
S
AGREEMENT
NUMBER
sg
SUBJECT
AGREEMENT
PERSON
3rd

Unification

- Two key roles:

Unification

- Two key roles:
  - Merge compatible feature structures

Unification

- Two key roles:
  - Merge compatible feature structures
  - Reject incompatible feature structures

Unification

- Two key roles:
  - Merge compatible feature structures
  - Reject incompatible feature structures
- Two structures can unify if

Unification

- Two key roles:
  - Merge compatible feature structures
  - Reject incompatible feature structures

- Two structures can unify if
  - Feature structures are identical
    - Result in same structure

Unification

- Two key roles:
  - Merge compatible feature structures
  - Reject incompatible feature structures

- Two structures can unify if
  - Feature structures are identical
    - Result in same structure
  - Feature structures match where both have values, differ in missing or underspecified
    - Resulting structure incorporates constraints of both

Subsumption

- Relation between feature structures
  - Less specific f.s. subsumes more specific f.s.
  - F.s. F subsumes f.s. G iff
    - For every feature x in F, F(x) subsumes G(x)
    - For all paths p and q in F s.t. F(p)=F(q), G(p)=G(q)

Subsumption

- Relation between feature structures
  - Less specific f.s. subsumes more specific f.s.
  - F.s. F subsumes f.s. G iff
    - For every feature x in F, F(x) subsumes G(x)
    - For all paths p and q in F s.t. F(p)=F(q), G(p)=G(q)
- Examples:
  - A: [Number SG], B: [Person 3]
  - C:[Number SG]
    - [Person 3]

Subsumption

- Relation between feature structures
  - Less specific f.s. subsumes more specific f.s.
  - F.s. F subsumes f.s. G iff
    - For every feature x in F, F(x) subsumes G(x)
    - For all paths p and q in F s.t. F(p)=F(q), G(p)=G(q)
- Examples:
  - A: [Number SG], B: [Person 3]
  - C:[Number SG]
    - [Person 3]
  - A subsumes C; B subsumes C; B,A don't subsume
    - Partial order on f.s.

Unification Examples

- Identical
  - [Number SG] U [Number SG]

Unification Examples

- Identical
  - [Number SG] U [Number SG]=[Number SG]
- Underspecified
  - [Number SG] U [Number []]

Unification Examples

- Identical
  - [Number SG] U [Number SG]=[Number SG]
- Underspecified
  - [Number SG] U [Number []] = [Number SG]
- Different specification
  - [Number SG] U [Person 3]

Unification Examples

- Identical
  - [Number SG] U [Number SG]=[Number SG]
- Underspecified
  - [Number SG] U [Number []] = [Number SG]
- Different specification
  - [Number SG] U [Person 3] = [Number SG]
  - [Person 3]
  - [Number SG] U [Number PL]

Unification Examples

- Identical
  - [Number SG] U [Number SG]=[Number SG]
- Underspecified
  - [Number SG] U [Number []] = [Number SG]
- Different specification
  - [Number SG] U [Person 3] = [Number SG]
  - [Person 3]
- Mismatched
  - [Number SG] U [Number PL] → Fails!

More Unification Examples

$$\begin{bmatrix} \text{AGREEMENT} & [1] \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [1] \end{bmatrix} \end{bmatrix} \quad \text{U}$$

$$\begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & 3 \\ \text{NUMBER} & \text{SG} \end{bmatrix} \end{bmatrix} \end{bmatrix} \quad =$$

$$\begin{bmatrix} \text{AGREEMENT} & [1] \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & [1] \begin{bmatrix} \text{PERSON} & 3 \\ \text{NUMBER} & \text{SG} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Features in CFGs: Agreement

- Goal:
    - Support agreement of NP/VP, Det Nominal
- Approach:
    - Augment CFG rules with features
    - Employ head features
        - Each phrase: VP, NP has head
            - Head: child that provides features to phrase
                - Associates grammatical role with word
                - VP – V; NP – Nom, etc

Agreement with Heads and Features

VP → Verb NP

NP → Det Nominal

Nominal → Noun

Noun → flights

Verb → serves

Agreement with Heads and Features

VP → Verb NP
<VP HEAD> = <Verb HEAD>

NP → Det Nominal

Nominal → Noun

Noun → flights

Verb → serves

Agreement with Heads and Features

VP → Verb NP
<VP HEAD> = <Verb HEAD>

NP → Det Nominal
<NP HEAD> = <Nominal HEAD>
<Det HEAD AGREEMENT> = <Nominal HEAD AGREEMENT>

Nominal → Noun

Noun → flights

Verb → serves

Agreement with Heads and Features

VP → Verb NP
<VP HEAD> = <Verb HEAD>

NP → Det Nominal
<NP HEAD> = <Nominal HEAD>
<Det HEAD AGREEMENT> = <Nominal HEAD AGREEMENT>

Nominal → Noun
<Nominal HEAD> = <Noun HEAD>

Noun → flights

Verb → serves

Agreement with Heads and Features

VP → Verb NP
<VP HEAD> = <Verb HEAD>

NP → Det Nominal
<NP HEAD> = <Nominal HEAD>
<Det HEAD AGREEMENT> = <Nominal HEAD AGREEMENT>

Nominal → Noun
<Nominal HEAD> = <Noun HEAD>

Noun → flights
<Noun HEAD AGREEMENT NUMBER> = PL

Verb → serves

Agreement with Heads and Features

VP → Verb NP
<VP HEAD> = <Verb HEAD>

NP → Det Nominal
<NP HEAD> = <Nominal HEAD>
<Det HEAD AGREEMENT> = <Nominal HEAD AGREEMENT>

Nominal → Noun
<Nominal HEAD> = <Noun HEAD>

Noun → flights
<Noun HEAD AGREEMENT NUMBER> = PL

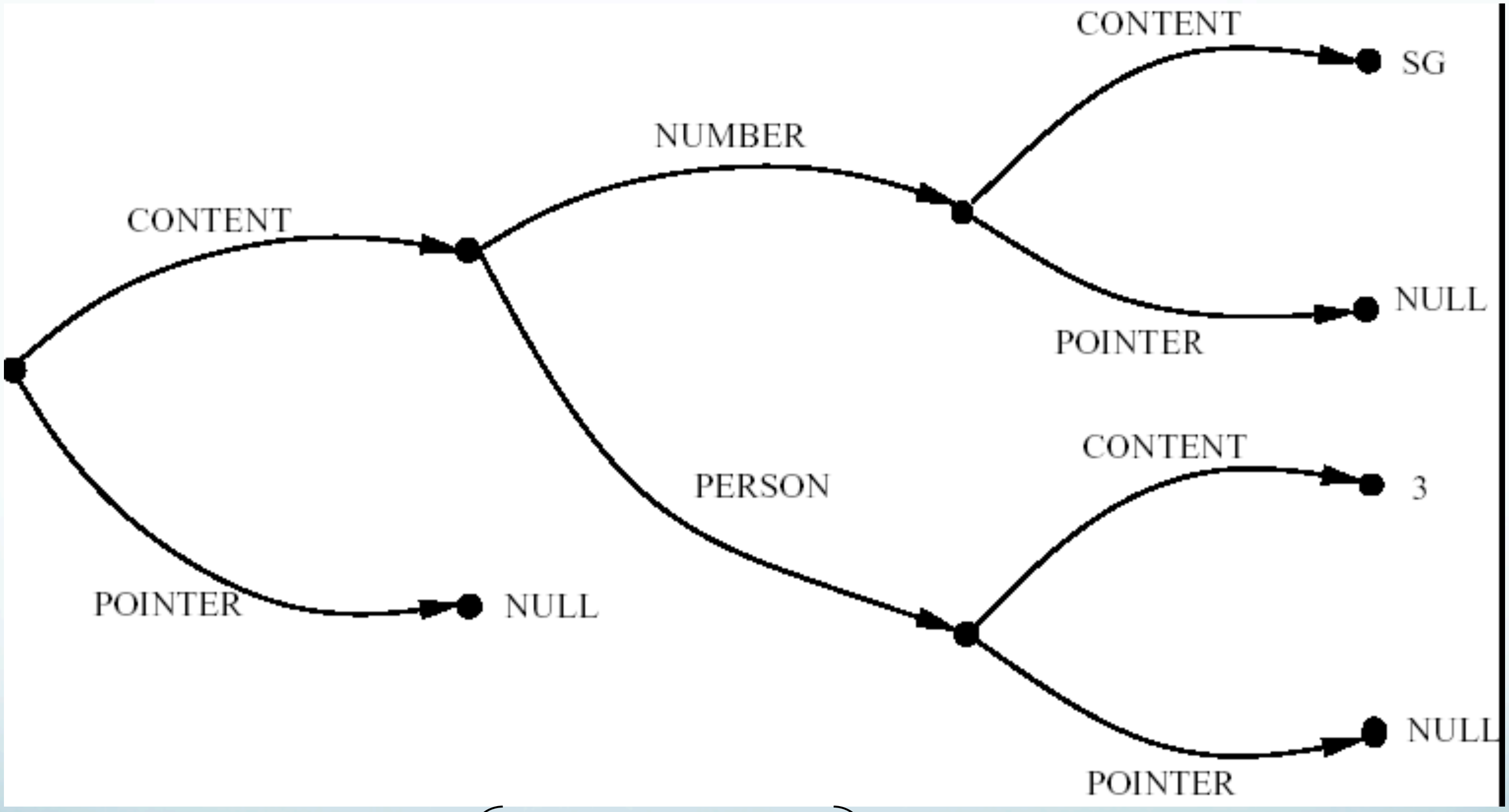
Verb → serves
<Verb HEAD AGREEMENT NUMBER> = SG
<Verb HEAD AGREEMENT PERSON> = 3

Feature Applications

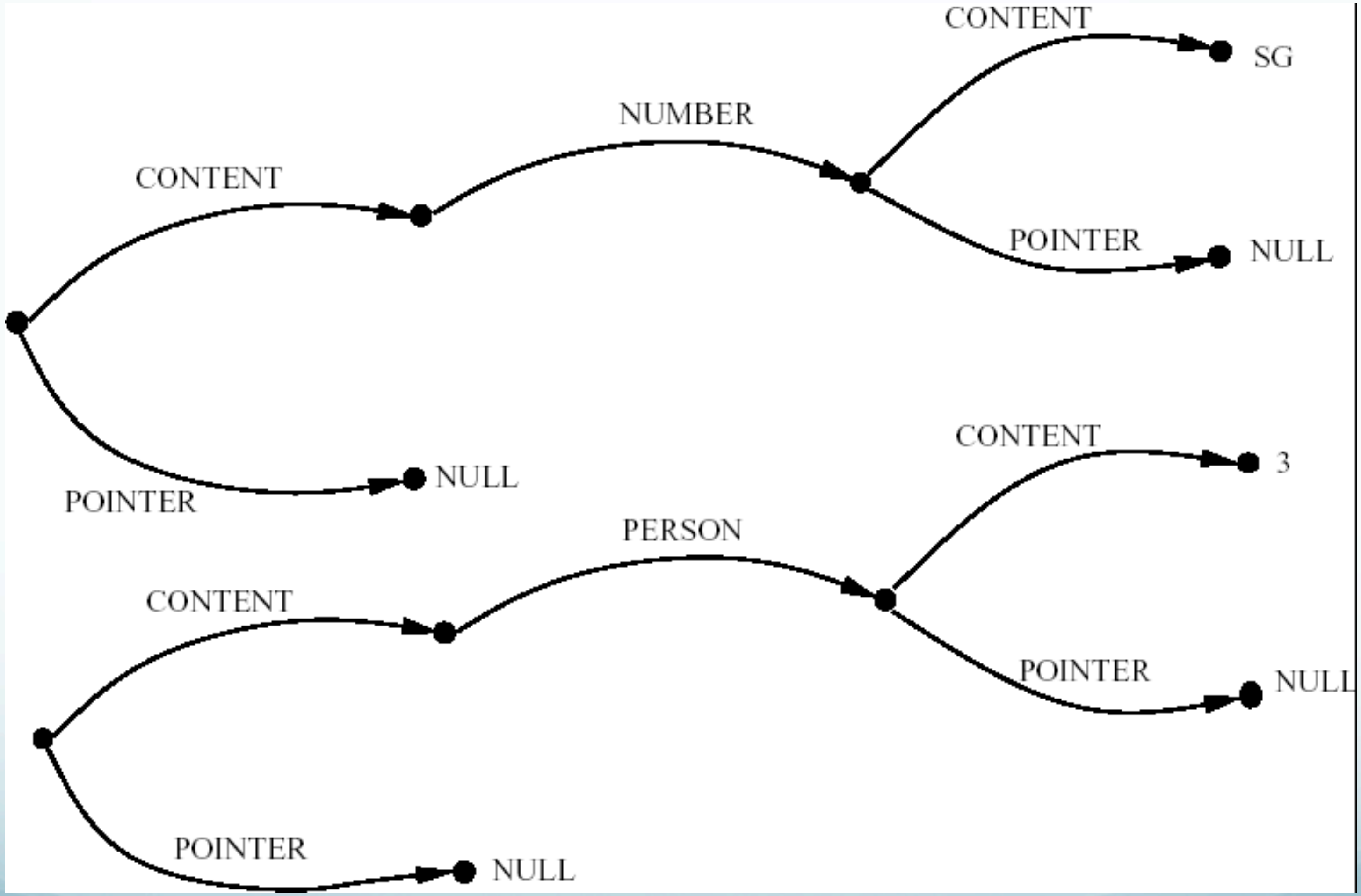
- Subcategorization:
  - Verb-Argument constraints
    - Number, type, characteristics of args (e.g. animate)
    - Also adjectives, nouns
- Long distance dependencies
  - E.g. filler-gap relations in wh-questions, rel

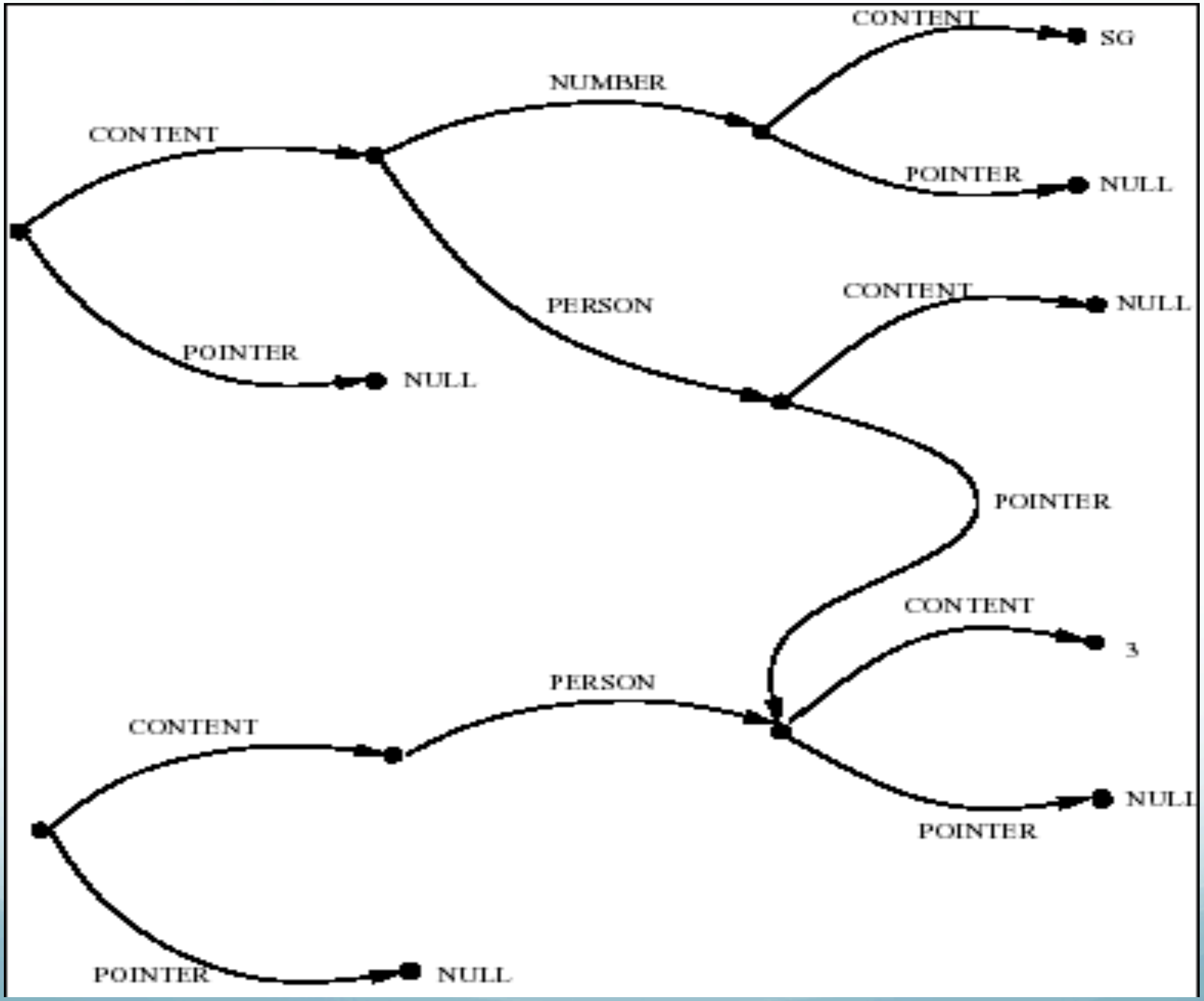
Implementing Unification

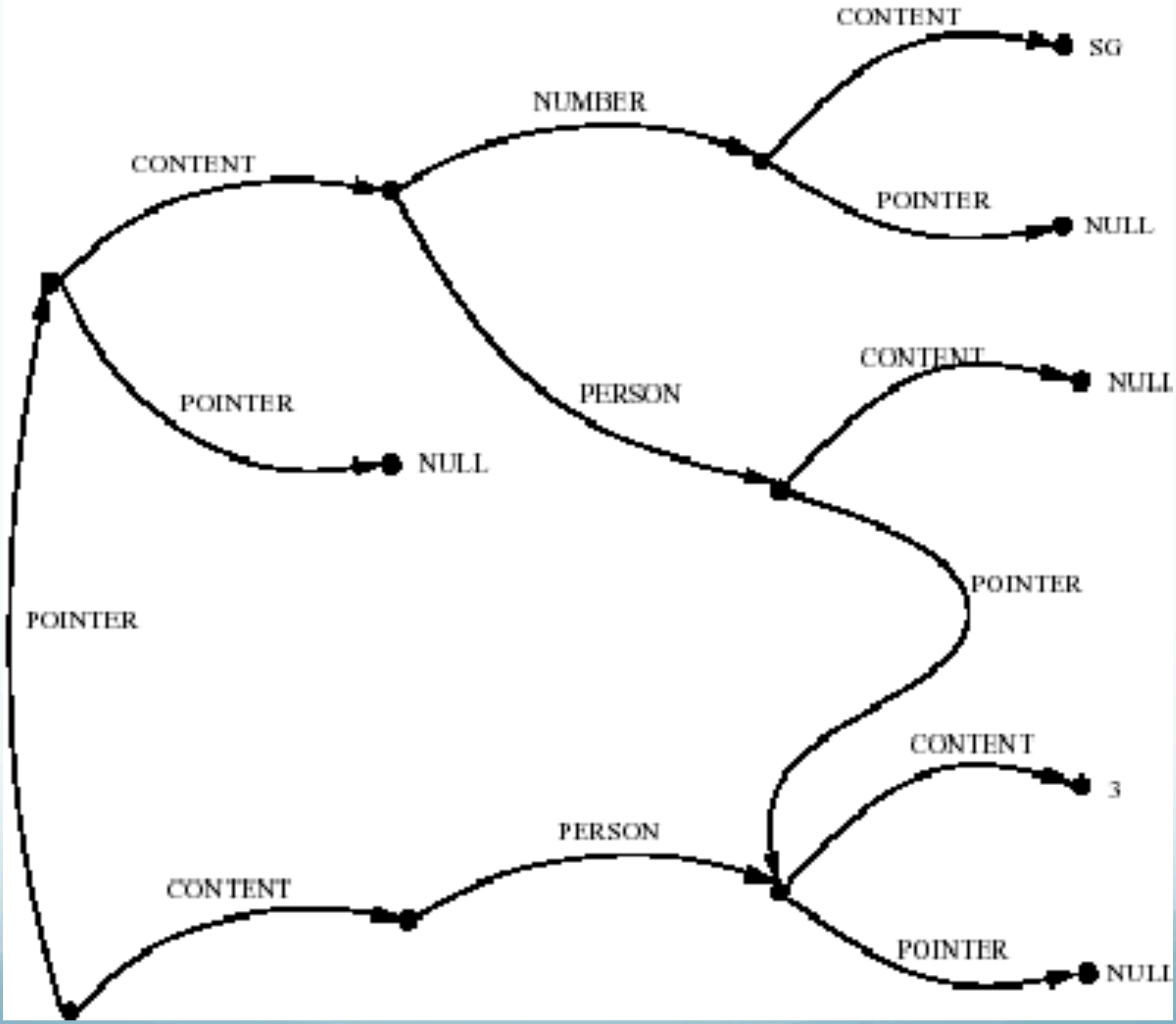
- Data Structure:
  - Extension of the DAG representation
  - Each f.s. has a content field and a pointer field
    - If pointer field is null, content field has the f.s.
    - If pointer field is non-null, it points to actual f.s.



$$\begin{bmatrix}\text{NUMBER} & \text{SG}\\ \text{PERSON} & 3\end{bmatrix}$$


CONTENT
SG
NUMBER
CONTENT
POINTER
NULL
POINTER
NULL
CONTENT
3
PERSON
CONTENT
POINTER
NULL
POINTER
NULL


CONTENT
SG
NUMBER
CONTENT
POINTER
NULL
PERSON
CONTENT
NULL
POINTER
NULL
POINTER
CONTENT
3
PERSON
CONTENT
NULL
POINTER
POINTER
NULL


CONTENT
SG
NUMBER
CONTENT
POINTER
NULL
CONTENT
NULL
PERSON
POINTER
NULL
POINTER
POINTER
CONTENT
3
PERSON
CONTENT
POINTER
NULL

Implementing Unification: II

- Algorithm:
  - Operates on pairs of feature structures
    - Order independent, destructive

Implementing Unification: II

- Algorithm:
    - Operates on pairs of feature structures
        - Order independent, destructive
    - If fs1 is null, point to fs2
    - If fs2 is null, point to fs1

Implementing Unification: II

- Algorithm:
  - Operates on pairs of feature structures
    - Order independent, destructive
  - If fs1 is null, point to fs2
  - If fs2 is null, point to fs1
  - If both are identical,

Implementing Unification: II

- Algorithm:
  - Operates on pairs of feature structures
    - Order independent, destructive
  - If fs1 is null, point to fs2
  - If fs2 is null, point to fs1
  - If both are identical, point fs1 to fs2, return fs2
    - Subsequent updates will update both
  - If non-identical atomic values

Implementing Unification: II

- Algorithm:
  - Operates on pairs of feature structures
    - Order independent, destructive
  - If fs1 is null, point to fs2
  - If fs2 is null, point to fs1
  - If both are identical, point fs1 to fs2, return fs2
    - Subsequent updates will update both
  - If non-identical atomic values, fail!

Implementing Unification: III

- If non-identical, complex structures
  - Recursively traverse all features of fs2
  - If feature in fs2 is missing in fs1
    - Add to fs1 with value null
  - If all unify, point fs2 to fs1 and return fs1

Example

$$\begin{bmatrix} \text{AGREEMENT [1]} & \begin{bmatrix} \text{NUMBER} & \text{SG} \\ \text{AGREEMENT [1]} & \end{bmatrix} \\ \text{SUBJECT} & \end{bmatrix} \text{U}$$

$$\begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

[AGREEMENT [1]] U [AGREEMENT [PERSON 3]]

[NUMBER SG] U [PERSON 3]

[NUMBER SG] U [PERSON 3]
[PERSON NULL]

Parsing with Features & Unification

- How can we integrate parsing with unification?
- Does unification have to happen at some particular point in parsing?

Parsing with Features & Unification

- How can we integrate parsing with unification?
- Does unification have to happen at some particular point in parsing?
  - Not really, unification is order-independent
- Simple strategy:

Parsing with Features & Unification

- How can we integrate parsing with unification?
- Does unification have to happen at some particular point in parsing?
  - Not really, unification is order-independent
- Simple strategy:
  - Run (any) parser, apply unification constraints
- Does it work

Parsing with Features & Unification

- How can we integrate parsing with unification?
- Does unification have to happen at some particular point in parsing?
  - Not really, unification is order-independent
- Simple strategy:
  - Run (any) parser, apply unification constraints
- Does it work? Yes
- Is it optimal?

Parsing with Features & Unification

- How can we integrate parsing with unification?
- Does unification have to happen at some particular point in parsing?
  - Not really, unification is order-independent
- Simple strategy:
  - Run (any) parser, apply unification constraints
- Does it work? Yes
- Is it optimal?
  - Not really, may construct lots of invalid parses

Unification and the Earley Parser

- Employ constraints to restrict addition to chart
- Actually pretty straightforward

Unification and the Earley Parser

- Employ constraints to restrict addition to chart
- Actually pretty straightforward
  - Augment rules with feature structure

Unification and the Earley Parser

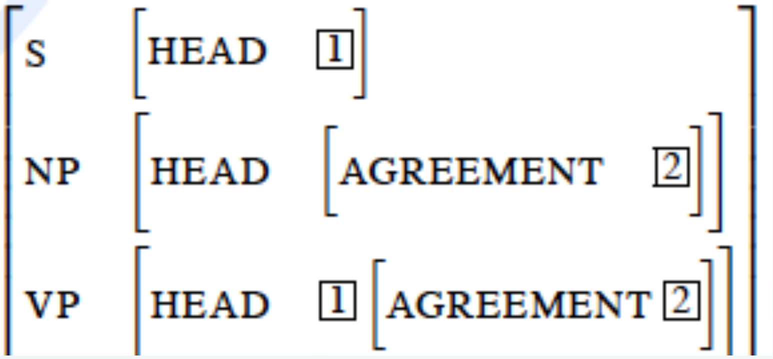
- Employ constraints to restrict addition to chart
- Actually pretty straightforward
  - Augment rules with feature structure
  - Augment state (chart entries) with DAG
    - Prediction adds DAG from rule

Unification and the Earley Parser

- Employ constraints to restrict addition to chart
- Actually pretty straightforward
  - Augment rules with feature structure
  - Augment state (chart entries) with DAG
    - Prediction adds DAG from rule
    - Completion applies unification (on copies)
      - Adds entry only if current DAG is NOT subsumed

Example Rule & State

- S → NP VP
  - <NP HEAD AGREEMENT> = <VP HEAD AGREEMENT>
  - <S HEAD> = <VP HEAD>

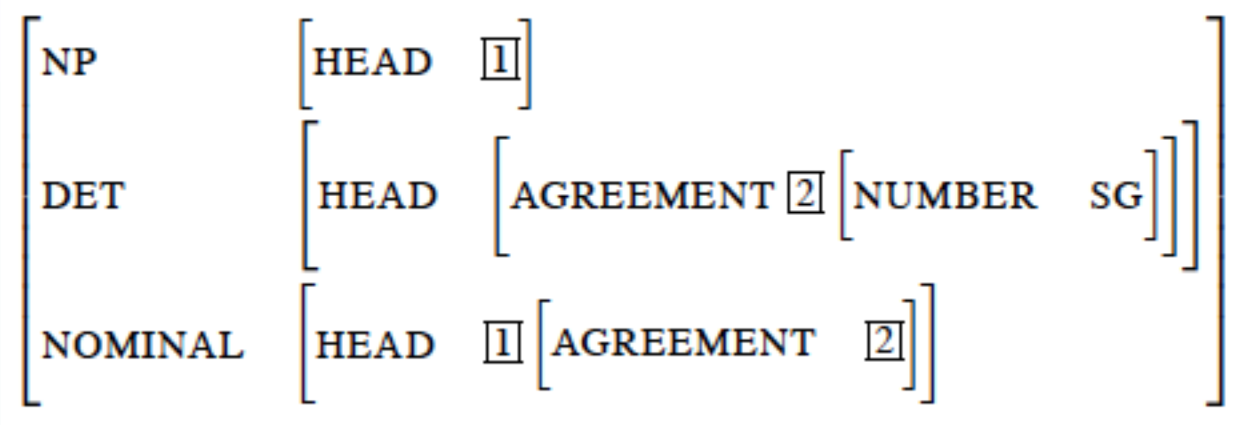


$$\begin{bmatrix} \text{S} & \begin{bmatrix} \text{HEAD} & \boxed{1} \end{bmatrix} \\ \text{NP} & \begin{bmatrix} \text{HEAD} & \begin{bmatrix} \text{AGREEMENT} & \boxed{2} \end{bmatrix} \end{bmatrix} \\ \text{VP} & \begin{bmatrix} \text{HEAD} & \boxed{1} \begin{bmatrix} \text{AGREEMENT} & \boxed{2} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

- Prediction: S→• NP VP, [0,0],[],Dag

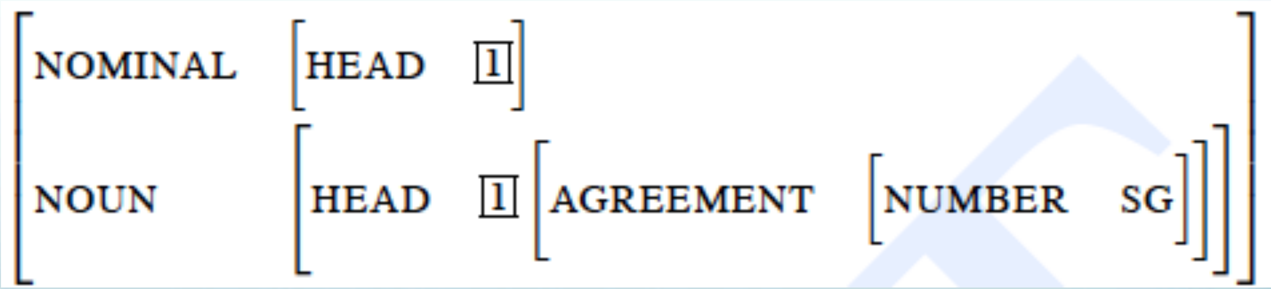
Example Completion

- Existing state: NP → Det • Nominal, [0,1],[S_{det}],Dag_1
- Dag_1:



$$\begin{bmatrix} \text{NP} & \begin{bmatrix}\text{HEAD} & \boxed{1}\end{bmatrix} \\ \text{DET} & \begin{bmatrix}\text{HEAD} & \begin{bmatrix}\text{AGREEMENT} & \boxed{2}\begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix}\end{bmatrix}\end{bmatrix} \\ \text{NOMINAL} & \begin{bmatrix}\text{HEAD} & \boxed{1}\begin{bmatrix}\text{AGREEMENT} & \boxed{2}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

- Completed state: Nominal → Noun•,[1,2],[S_{noun}],Dag_2
- Dag_2:



$$\begin{bmatrix} \text{NOMINAL} & \begin{bmatrix}\text{HEAD} & \boxed{1}\end{bmatrix} \\ \text{NOUN} & \begin{bmatrix}\text{HEAD} & \boxed{1}\begin{bmatrix}\text{AGREEMENT} & \begin{bmatrix}\text{NUMBER} & \text{SG}\end{bmatrix}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

Unification Parsing

- Abstracts over categories
  - S → NP VP →
    - X0 → X1 X2; <X0 cat> = S; <X1 cat>=NP;
    - <X2 cat>=VP
  - Conjunction:
    - X0 → X1 and X2; <X1 cat> = <X2 cat>;
    - <X0 cat>=<X1 cat>
- Issue: Completer depends on categories
- Solution: Completer looks for DAGs which unify with the just-completed state' s DAG

Extensions

- Types and inheritance
  - Issue: generalization across feature structures
    - E.g. many variants of agreement
      - More or less specific: 3rd vs sg vs 3rdsg

Extensions

- Types and inheritance
  - Issue: generalization across feature structures
    - E.g. many variants of agreement
      - More or less specific: 3rd vs sg vs 3rdsg
  - Approach: Type hierarchy
    - Simple atomic types match literally
    - Multiple inheritance hierarchy
      - Unification of subtypes is most general type that is more specific than two input types
    - Complex types encode legal features, etc

Conclusion

- Features allow encoding of constraints
  - Enables compact representation of rules
  - Supports natural generalizations

- Unification ensures compatibility of features
  - Integrates easily with existing parsing mech.

- Many unification-based grammatical theories