Dependency Parsing
&
Feature-based Parsing

Ling 571 — Deep Processing Techniques for NLP
October 22, 2018
Ryan Georgi

$$P\left(\lim_{x\to 0}\frac{x}{n}\right) \neq P(0)$$

Fun Fact:
No statistician has ever been on an airplane.

$$P(ADJ \rightarrow \text{‘brillig’}) = 1.2\times10^{-342}$$

(Think “[Jabberwocky]”)

$$P(ADJ \rightarrow \text{‘brilliger’}) \neq 0$$

…I just used it!

Input query:	*“Is ‘brilliger' an adjective?”*
Parser with OOV handling:	*“Err… probably not?”*

Source: Saturday Morning Breakfast Cereal

HW #4 Follow-up

HW #4 Follow-up: OOV Handling

- As we discussed previously, you will find OOV tokens
- Sometimes this as as simple as case-sensitivity:

OOV: Case Sensitivity

Sentence #23: "*Arriving before four p.m .*"

```
  ----------------------------------------------------------------------------------------------------------------
  |   |                         |                                   |                                    |
0 ----------------------------------------------------------------------------------------------------------------
    | IN -> "before" [-3.8326] |                                   | PP -> 1•IN•2 2•NP•4 [-13.9845]      | TOP -> 1•PP•4 4•PUNC•5 [-19.4677]       |
    |                          |                                   | FRAG_PP -> 1•IN•2 2•NP•4 [-13.1613] | TOP -> 1•FRAG_PP•4 4•PUNC•5 [-18.6445] |
  1 --------------------------------------------------------------------------------------------------------------
                               | CD -> "four" [-4.3438] | PRIME -> 2•CD•3 3•RB•4 [-10.3372]        | TOP -> 2•NP•4 4•PUNC•5 [-11.4025]       |
                               |                        | NP_PRIME -> 2•CD•3 3•RB•4 [-10.2784]     |                                        |
                               |                        | NP -> 2•CD•3 3•RB•4 [-8.9233]            |                                        |
                             2 ----------------------------------------------------------------------------------
                                                        | RB -> "p.m" [-1.1144]                    |                                        |
                                                      3 ---------------------------------------------------------
                                                                                                   | PUNC -> "." [-0.3396]                  |
                                                                                                 4 ----------------
                                                                                                                  5
```

"**a**rriving" is in our grammar, but not "**A**rriving"

OOV: Case Sensitivity

Sentence #23: "*Arriving before four p.m .*"

```
-----------------------------------------------------------------------------------------------------------------------------------------------------------
| VBG -> "arriving" [-1.0372]            |                          |                          | PRIME -> 0•VBG•1 1•PP•4 [-19.6776]           | TOP -> 0•FRAG_VP•4 4•PUNC•5 [-21.1981] |
| VP_VBG -> "arriving" [-0.6931]         |                          |                          | VP_PRIME -> 0•VBG•1 1•PP•4 [-18.0049]        | TOP -> 0•VP•4 4•PUNC•5 [-20.1503]      |
| S_VP_VBG -> "arriving" [0.0000]        |                          |                          | VP -> 0•VBG•1 1•PP•4 [-17.6629]              |                                        |
|                                        |                          |                          | FRAG_VP -> 0•VBG•1 1•PP•4 [-16.2257]         |                                        |
|                                        |                          |                          | FRAG_VP_PRIME -> 0•VBG•1 1•PP•4 [-15.8691]   |                                        |
0 ---------------------------------------------------------------------------------------------------------------------------------------------------------
                                         | IN -> "before" [-3.8326] |                          | PP -> 1•IN•2 2•NP•4 [-13.9845]               | TOP -> 1•PP•4 4•PUNC•5 [-19.4677]      |
                                         |                          |                          | FRAG_PP -> 1•IN•2 2•NP•4 [-13.1613]          | TOP -> 1•FRAG_PP•4 4•PUNC•5 [-18.6445] |
                                       1 ---------------------------------------------------------------------------------------------------------------
                                                                    | CD -> "four" [-4.3438]   | PRIME -> 2•CD•3 3•RB•4 [-10.3372]            | TOP -> 2•NP•4 4•PUNC•5 [-11.4025]      |
                                                                    |                          | NP_PRIME -> 2•CD•3 3•RB•4 [-10.2784]         |                                        |
                                                                    |                          | NP -> 2•CD•3 3•RB•4 [-8.9233]                |                                        |
                                                                  2 --------------------------------------------------------------------------------------
                                                                                               | RB -> "p.m" [-1.1144]                        |                                        |
                                                                                             3 -----------------------------------------------------------
                                                                                                                                              | PUNC -> "." [-0.3396]                  |
                                                                                                                                            4 ------------------------------------------
                                                                                                                                                                                       5
```

HW #4 Follow-up: OOV Handling

- Propose some number of N most likely tags at runtime…

OOV: Propose POS Tags

"Show me Ground transportation in Denver during weekdays ." — No "during"!

	FRAG_NP_PRIME → 2FRAG_NP_PRIME 4 PP 6[-21.810] FRAG_NP → 2FRAG_NP_PRIME 4 PP 6[-20.858]			
	NP_PRIME → 3 NN 4 PP 6[-16.296] PRIME → 3 NN 4 PP 6[-15.949]			
IN → "in" [-2.4018]	PP → 4 IN 5 NP_NNP 6[-7.505] FRAG_PP → 4 IN 5NP_NNP 6 [-6.828]			
5	NNP → "Denver" [-4.4002] NP_NNP → "Denver" [-3.3280]			
	6			
		7	NNS → "weekdays" [-5.5759] NP_NNS → "weekdays" [-3.7257]	TOP → 7NP_NNS 8PUNC 9[-11.001]
			8	PUNC → "." [-0.3396]
				9

OOV: Propose POS Tags

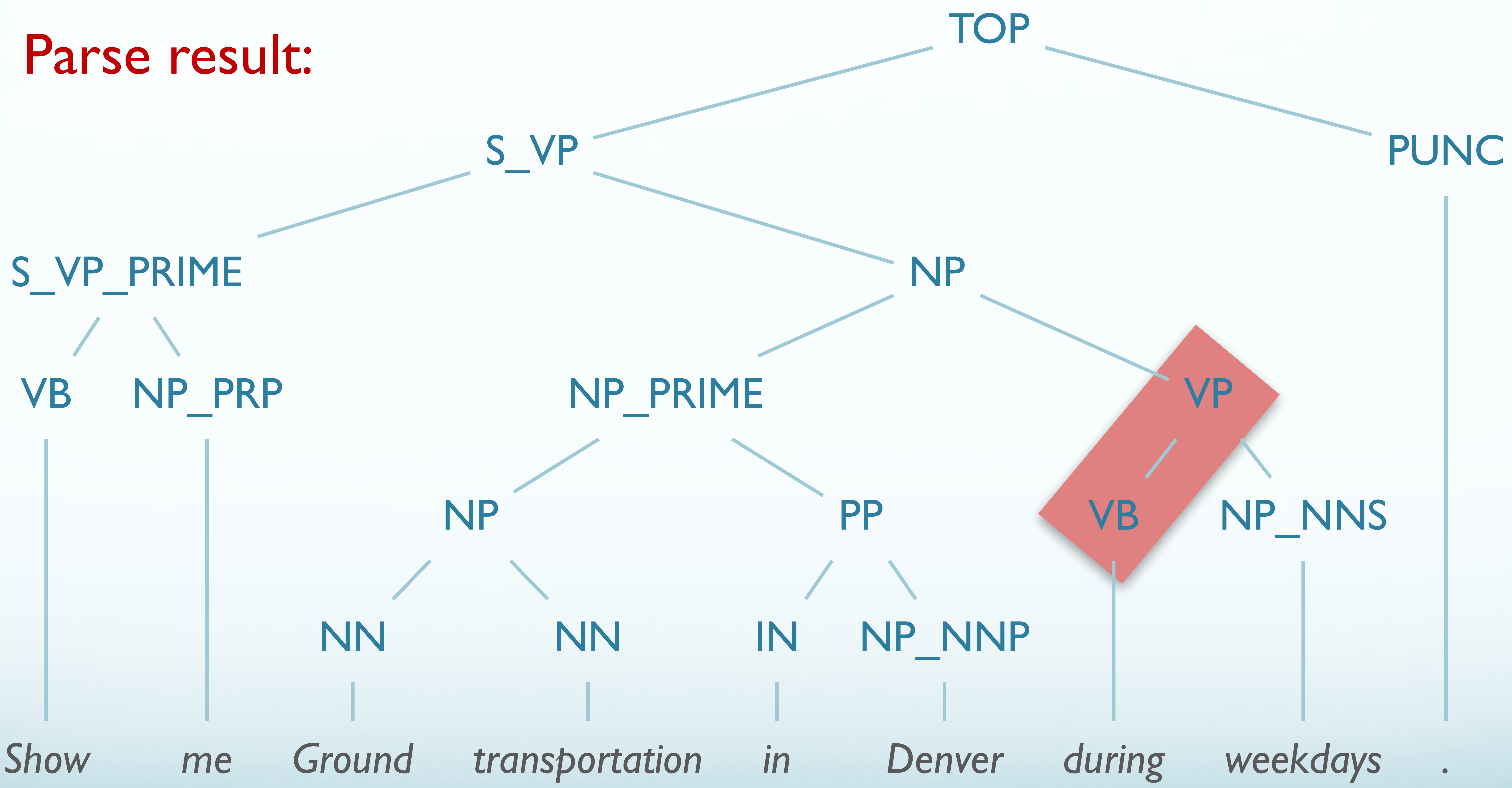
“Show me Ground transportation in Denver during weekdays .” — No “during”!

FRAG_NP_PRIME → … FRAG_NP → …	FRAG_NP_PRIME → … FRAG_NP → …	FRAG_NP →… FRAG_NP → …	TOP → 2FRAG_NP 8 PUNC 9[-34.939] TOP → 2FRAG_NP 8 PUNC 9[-34.006]
NP_PRIME → … PRIME → …	PRIME → 3 NN 4PP 7 [-17.145] QP → 3 PRIME 6CD 7 [-15.930]	NP → 3 PRIME 7NNS 8 [-26.542] NP → 3 QP 7 NNS 8 [-26.398]	TOP → 3NP 8PUNC 9[-29.022] TOP → 3NP 8PUNC 9[-28.877]
PP → … FRAG_PP → …	PP → 4 IN 5 NP 7[-8.701] FRAG_PP → 4 IN 5NP 7 [-7.878]	PP → 4 IN 5 NP 8[-19.056] FRAG_PP → 4 IN 5NP 8 [-18.233]	TOP → 4PP 8PUNC 9[-24.540] TOP → 4FRAG_PP 8 PUNC 9[-23.716]
NNP → "Denver" [-4.4002] NP_NNP → "Denver" [-3.3280]	NP_PRIME → 5NNP 6 NNP 7[-6.110] NP → 5 NNP 6NNP 7 [-5.070]	NP → 5 NP 7 NNS 8 [-17.330] NP → 5NP_PRIME 7 NNS 8 [-15.426]	TOP → 5NP 8PUNC 9[-19.809] TOP → 5NP 8PUNC 9[-17.905]
6	NNP → "during" [1.0000] NN → "during" [1.0000] NP_NNP → "during" [1.0000] VB → "during" [1.0000] CD → "during" [1.0000]	VP → 6 VB 7NP_NNS 8[-8.922] S_VP → 6 VB 7NP_NNS 8[-6.611]	TOP → 6VP 8PUNC 9[-11.410] TOP → 6S_VP 8PUNC 9[-9.176]
	7	NNS → "weekdays" [-5.5759] NP_NNS → "weekdays" [-3.7257]	TOP → 7NP_NNS 8 PUNC 9[-11.001]
		8	PUNC → "." [-0.3396]
			9

OOV: Propose POS Tags

"Show me Ground transportation in Denver during weekdays ." — No "during"!

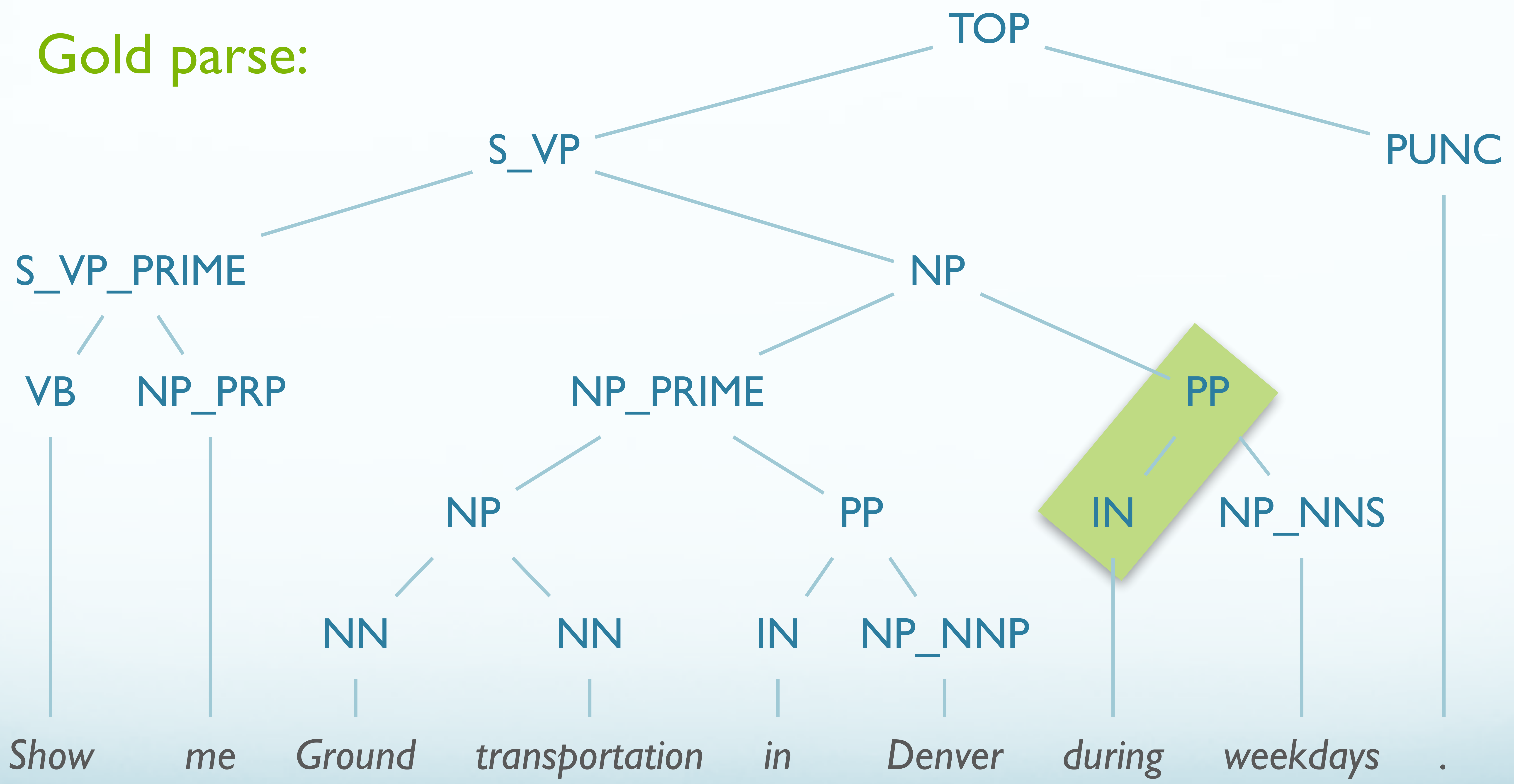
Parse result:



OOV: Propose POS Tags

"Show me Ground transportation in Denver during weekdays ." — No "during"!

Gold parse:



Problems with this approach?

Handling OOV

- **Option #1**:
  - Choose subset of training data vocab to be hidden
  - Hidden words replaced by `<UNK>`
  - Run induction as usual, but some words are now '`<UNK>`'
- **Option #2**:
  - Replace first occurrence of every word with `<UNK>`
- (See J&M 2nd ed 4.3.2 — 3rd ed, 3.3.1)

Problems with These Approaches?

- **Option #1**
  - May sample "closed-class" words
  - Closed-class words are disproportionately more common
    - ∴ Approximation will be worse the more data there is, *because Zipf*
- **Option #2**
  - **Con**: Requires a lot more data
  - **Pros**: Samples from all word classes
    - Will only count closed-class words once

HW #4 Extra Credit Opportunity

- ***Up to 10 points***:
  - Design an OOV treatment for handling treebank training data that:
    1. Uses `<UNK>` token sampling
    2. Is smart about open vs. closed-class words
  - You can modify reference code for HW #4.

Other Announcements

Other Announcements

- HW #2
  - Expect grades by EOD
- HW #3
  - By Wednesday, most likely.

Today

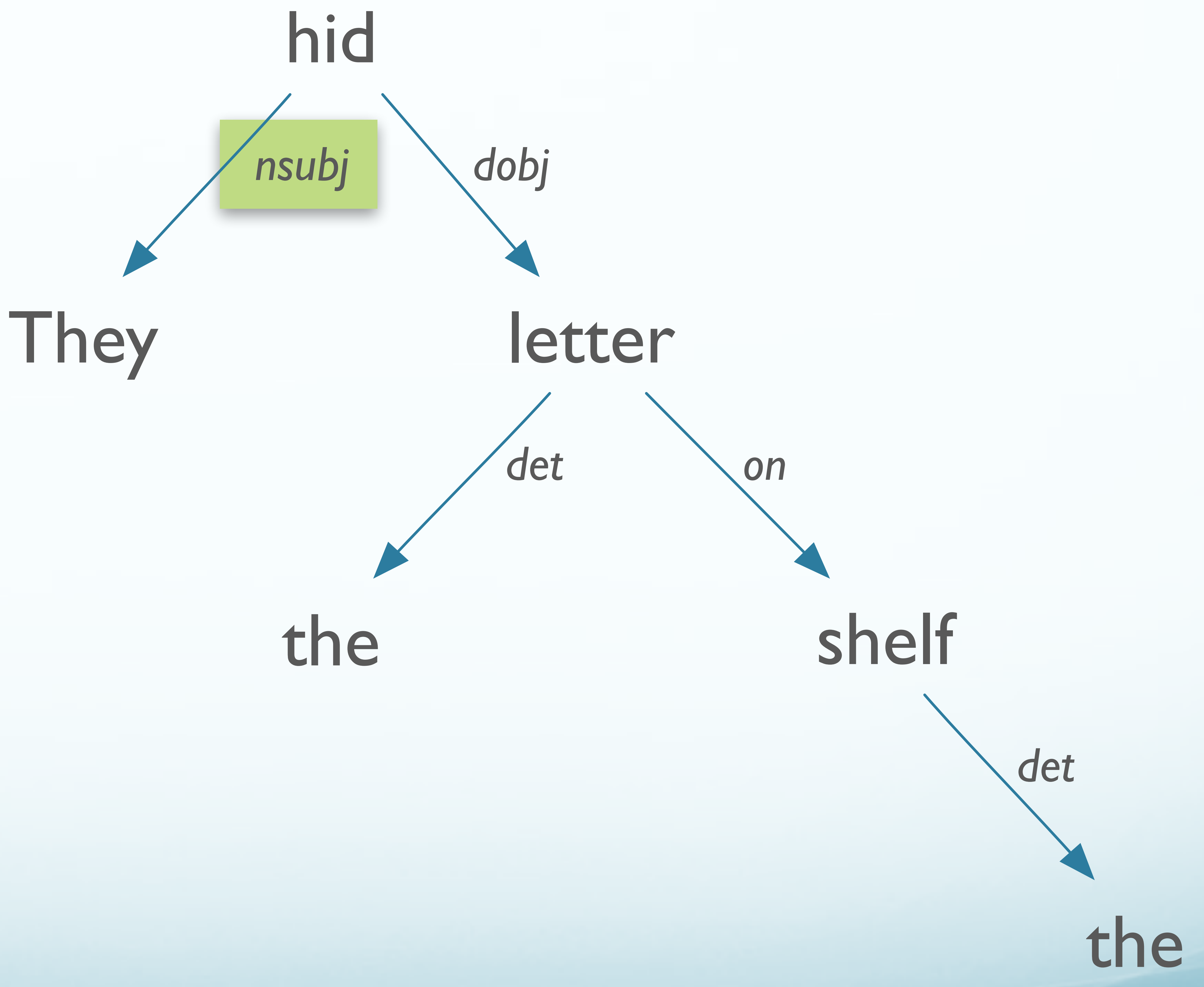
- **Dependency Parsing**
  - Transition-based Parsing
- Feature-based Parsing
  - Motivation
  - Features
  - Unification

Dependency Parse Example:

They hid the letter on the shelf

Argument Dependencies	
Abbreviation	**Description**
nsubj	nominal subject
csubj	clausal subject
dobj	direct object
iobj	indirect object
pobj	object of preposition

Modifier Dependencies	
Abbreviation	**Description**
tmod	temporal modifier
appos	appositional modifier
det	determiner
prep	prepositional modifier



Transition-Based Parsing

- Parsing defined in terms of sequence of transitions
- Alternative methods for learning/decoding
  - Most common model: Greedy classification-based approach
  - Very efficient: $O(n)$
- Best-known implementations:
  - Nivre's MALTParser
    - Nivre et al (2006); Nivre & Hall (2007)

Transition-Based Parsing

- A transition-based system for dependency parsing is:
  - A set of **configurations** C
  - A set of **transitions** between configurations
  - A transition function between configurations
  - An initialization function (for C_0)
  - A set of terminal configurations ("end states")

Configurations

- A configuration for a sentence x is the triple (Σ, B, A):
- Σ is a stack with elements corresponding to the nodes (words + $\texttt{ROOT}$) in x
- B (aka the buffer) is a list of nodes in x
- A is the set of dependency arcs in the analysis so far,
  - (w_i, L, w_j), where w_x is a node in x and L is a dependency label

Transitions

- Transitions convert one configuration to another
  - $C_i = t(C_i\text{-}1)$, where t is the transition
- Dependency graph for a sent:
  - The set of arcs resulting from a sequence of transitions
- The parse of the sentence is that resulting from the initial state through the sequence of transitions to a legal terminal state

Dependencies → Transitions

- To parse a sentence, we need the sequence of transitions that derives it
- How can we determine sequence of transitions, given a parse?
- This is defining our ***oracle*** function:
  - How to take a parse and translate it into a series of transitions

Dependencies → Transitions

- Many different oracles:
  - Nivre's arc-standard
  - Nivre's arc-eager
  - Non-projectivity with Attardi's
  - …
- Generally:
  - Use oracle to identify gold transitions
  - Train **classifier** to predict best transition in new config

Nivre's Arc-Standard Oracle

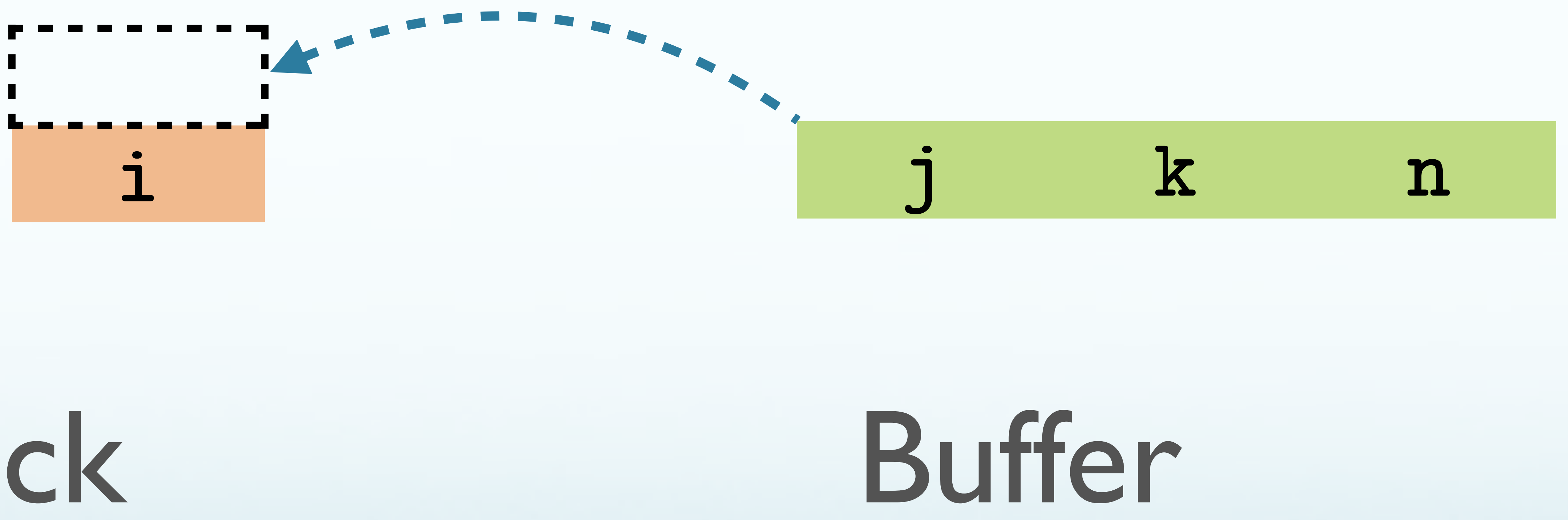
- Words: $w_1, \ldots, w_n$
  - w_0 = ROOT
- Initialization:
  - Stack = $[w_0]$; Buffer = $[w_1, \ldots w_n]$; Arcs = $\varnothing$
- Termination:
  - Stack = σ; Buffer= [ ]; Arcs = A
    - for any σ and A

Nivre's Arc-Standard Oracle

- Transitions are one of three:
  - Shift
  - Left-Arc
  - Right-Arc

Transitions: Shift

- *Shift* first element of buffer to top of stack.
  - [i] [j,k,n] [] → [i,j] [k,…,n] []



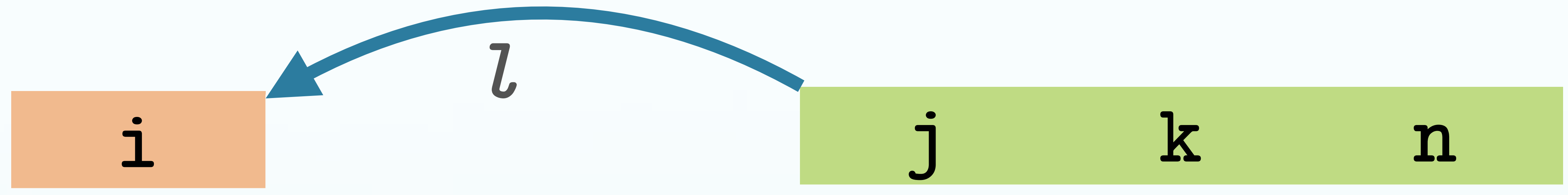
Transitions: Shift

- *Shift* first element of buffer to top of stack.
  - [i][j,k,n][] → [i,j][k,…,n][]

Stack Buffer Arcs

Transitions: Left-Arc

- Add arc from first element of buffer `j` to element at top of stack `i` with dependency label `l`
  - Pop `i` from stack.
  - `[i] [j,k,n] A → [i] [k,…,n] A∪[(j,l,i)]`



Stack Buffer Arcs

Transitions: Left-Arc

- Add arc from first element of buffer `j` to element at top of stack `i` with dependency label `l`
  - Pop `i` from stack.
  - `[i] [j,k,n] A → [i] [k,…,n] A∪[(j,l,i)]`

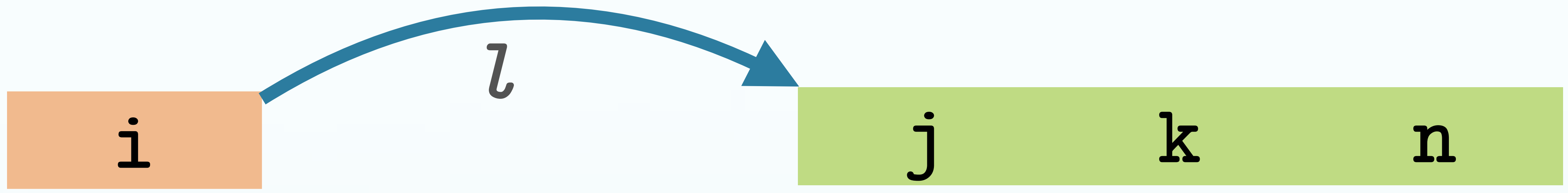
j	k	n

(j,l,i)

Stack Buffer Arcs

Transitions: Right-Arc

- Add arc from top of stack `i` to first element of buffer `j` with dependency label `l`
  - Replace j with i as front of buffer; pop j from stack.
  - `[i] [j,k,n] A → [i] [k,…,n] A∪[(j,l,i)]`



Stack Buffer Arcs

Transitions: Right-Arc

- Add arc from top of stack `i` to first element of buffer `j` with dependency label `l`
  - Replace j with i as front of buffer; pop j from stack.
  - `[i] [j,k,n] A → [i] [k,…,n] A∪[(j,l,i)]`

j

i k n

Stack

Buffer

Arcs

Transitions: Right-Arc

- Add arc from top of stack `i` to first element of buffer `j` with dependency label `l`
  - Replace j with i as front of buffer; pop j from stack.
  - `[i] [j,k,n] A → [i] [k,…,n] A∪[(j,l,i)]`

Stack	Buffer	Arcs
	`i k n`	`(i,l,j)`

Training Process

- Each step of the algorithm is a decision point between the three states
- We want to train a model to decide between the three options at each step
  - (Reduce to a classification problem)
- We start with:
  - A treebank
  - An *oracle* process for guiding the transitions
  - A discriminative learner to relate the transition to features of the current configuration

Training Process, Formally:

(Σ, B, A)

```
1) c ← c0(S)
2) while c is not terminal
3)     t ← o(c)  # Choose the (o)ptimal transition for the config c
4)     c ← t(c)  # Move to the next configuration
5) return Gc
```

Testing Process, Formally:

(Σ, B, A)

$c \leftarrow c_{\emptyset}(S)$

while c is not terminal

    $t \leftarrow \lambda_c(c)$ # Choose the transition given model parameters at c

    $c \leftarrow t(c)$ # Move to the next configuration

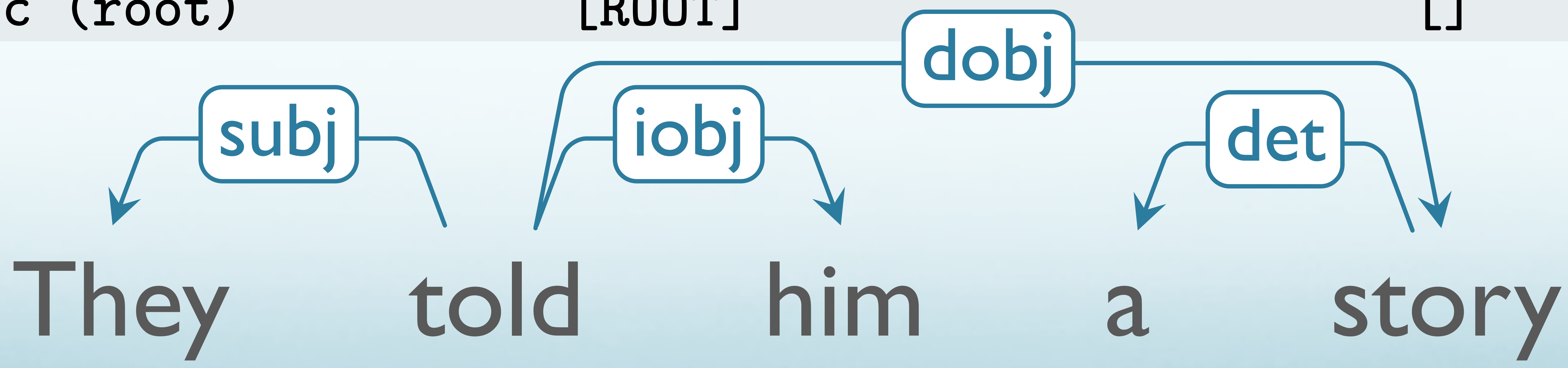
return G_c

Representing Configurations with Features

- ***Address***
  - Locate a given word:
    - By position in stack
    - By position in buffer
    - By attachment to a word in buffer
- ***Attributes***
  - Identity of word
  - lemma for word
  - POS tag of word
  - Dependency label for word ← *conditioned on previous decisions!*

Example: (Ballesteros et al 2015)

Action	Stack	Buffer
	[ROOT]	[They told him a story]
Shift	[ROOT, They]	[told him a story]
Left-Arc (subj)	[ROOT, told]	[him a story]
Shift	[ROOT, told, him]	[a story]
Right-Arc (iobj)	[ROOT, told]	[a story]
Shift	[ROOT, told, a]	[story]
Shift	[ROOT,told, a, story]	[]
Left-Arc (Det)	[ROOT, told, story]	[]
Right-Arc (dobj)	[ROOT, told]	[]
Right-Arc (root)	[ROOT]	[]



Transition-Based Parsing Summary

- *Shift-Reduce* paradigm, bottom-up approach
- ***Pros***:
  - Single pass, $O(n)$ complexity
  - Reduce parsing to classification problem; easy to introduce new features
- ***Cons***:
  - Only makes local decisions, may not find global optimum
  - Does not handle non-projective trees without hacks
    - e.g. transforming nonprojective trees to projective in training data; reconverting after

Other Notes

- …is this a parser?
  - No, not really!
  - Transforms problem into sequence labeling task, of a sort.
    - e.g. (`SH, LA, SH, RA, SH, SH, LA, RA`)
    - Sequence score is sum of transition scores
- Classifier: Any
  - Originally, SVMs
  - Currently: NNs + LSTMs
- State-of-the-art: UAS: 92.5%; LAS: 90.5%

Dependency Parsing: Summary

- Dependency Grammars:
  - Compactly represent pred–arg structure
  - Lexicalized, localized
  - Natural handling of flexible word order
- Dependency parsing:
  - Conversion to phrase structure trees
  - Graph-based parsing (MST), efficient non-proj $\mathbf{O}(n^2)$
  - Transition-based parser
    - MALTparser: very efficient $\mathbf{O}(n)$
      - Optimizes local decisions based on many rich features

Roadmap

- Dependency Parsing
  - Transition-based Parsing
- **Feature-based Parsing**
  - Motivation
  - Features
  - Unification

Feature-Based Parsing

Constraints & Compactness

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

Enforcing Constraints with CFG Rules

- Agreement
  - $S \rightarrow NP_{sg+3p}\ VP_{sg+3p}$
  - $S \rightarrow NP_{pl+3p}\ VP_{pl+3p}$
- Subcategorization:
  - $VP \rightarrow V_{transitive}\ NP$
  - $VP \rightarrow V_{intransitive}$
  - $VP \rightarrow V_{ditransitive}\ NP\ NP$
- Explosive, and loses key generalizations

Feature Grammars

- Need compact, general constraint
- S → NP VP [iff NP and VP agree]
- How can we describe agreement & subcategory?
  - Decompose into elementary features that must be consistent
    - e.g. Agreement on 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 structure" — sharing a structure
- Represented as
  - Attribute-Value Matrix (AVM)
  - Directed Acyclic Graph (DAG)

Attribute-Value Matrices (AVMs)

$$\begin{bmatrix} \text{ATTRIBUTE}_1 & \mathit{value}_1 \\ \text{ATTRIBUTE}_2 & \mathit{value}_2 \\ \vdots & \\ \text{ATTRIBUTE}_n & \mathit{value}_n \end{bmatrix}$$

AVM Examples

(A) $\begin{bmatrix} \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{bmatrix}$

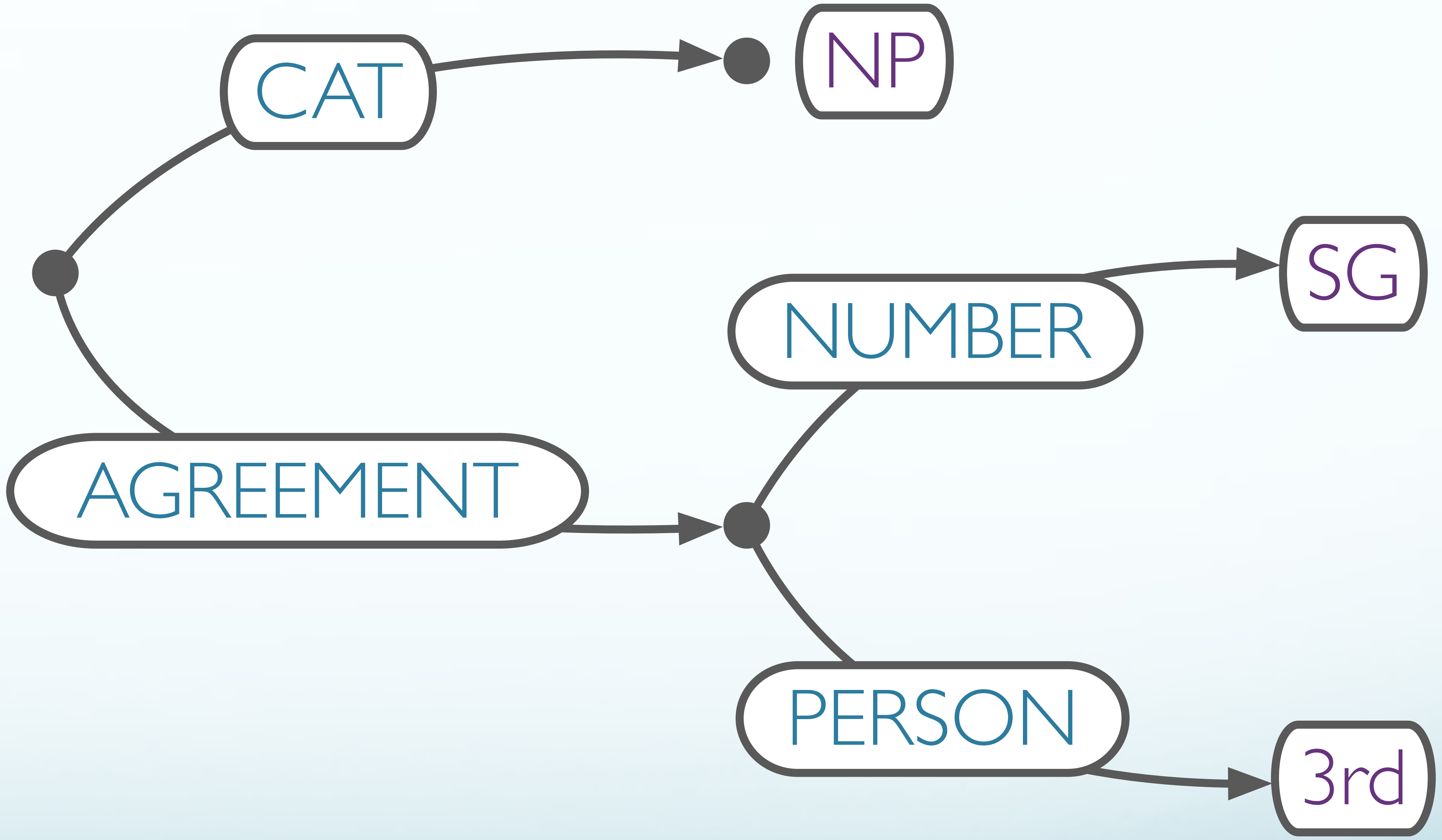
(B) $\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{bmatrix}$

(C) $\begin{bmatrix} \text{CAT} & \text{NP} \\ \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{bmatrix} \end{bmatrix}$

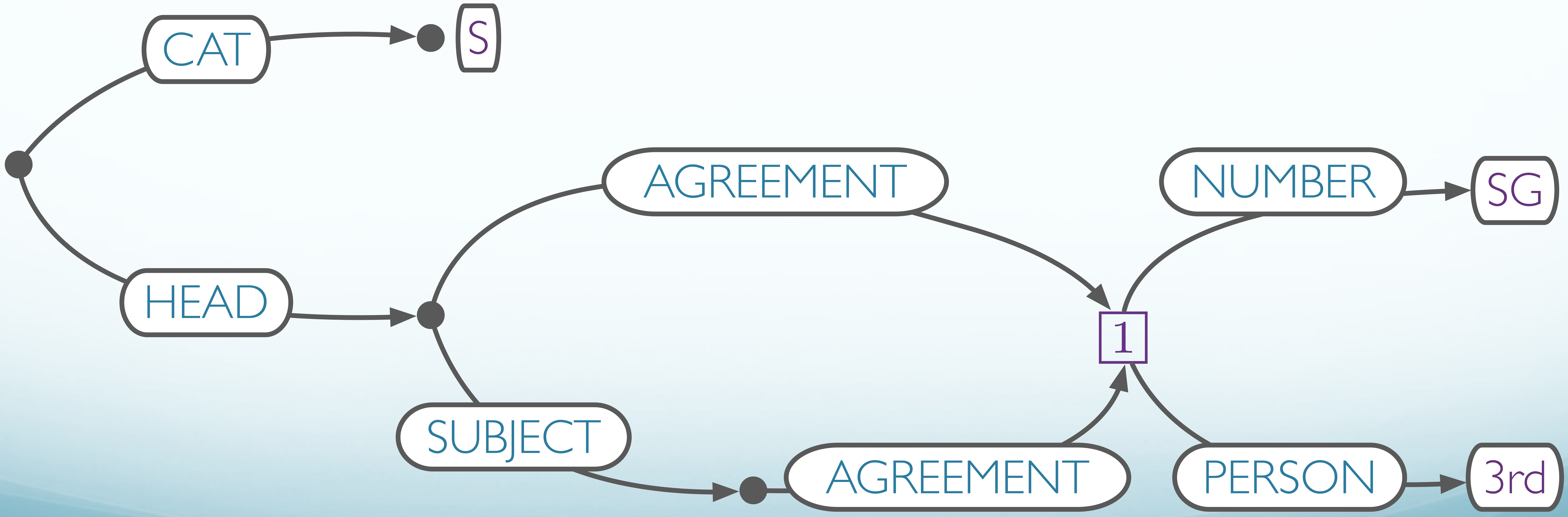
(D) $\begin{bmatrix} \text{CAT} & \text{S} \\ \text{HEAD} & \begin{bmatrix} \text{AGREEMENT} \ \boxed{1} & \begin{bmatrix} \text{NUMBER} & \text{PL} \\ \text{PERSON} & 3 \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} \ \boxed{1} \end{bmatrix} \end{bmatrix} \end{bmatrix}$

AVM vs. DAG

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



$$\begin{bmatrix} \text{CAT} & \text{S} \\ \text{HEAD} & \begin{bmatrix} \text{AGREEMENT} & \boxed{1}\begin{bmatrix}\text{NUMBER} & \text{PL}\\ \text{PERSON} & 3\end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix}\text{AGREEMENT} & \boxed{1}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$



Using Feature Structures

- Feature Structures provide formalism to specify constraints
- …but how to apply the constraints?
- ***Unification***

Unification: $\sqcup$

- Two key roles:
  - Merge compatible feature structures
  - Reject incompatible feature structures
- Two structures can unify if:
  - Feature structures ***match where both have values***
  - Feature structures ***differ only where one value is missing or underspecified***
    - Missing or underspecified values are filled with constraints of other
- Result of unification incorporates constraints of both

Subsumption

- Less specific feature structure ***subsumes*** more specific feature structure
- FS F subsubmes FS 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 = $\left[\text{NUMBER SG}\right]$ B = $\left[\text{PERSON 3}\right]$

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

- A *subsumes* C
- B *subsumes* C
- B & A *don't subsume*

Unification Examples

- Identical $\begin{bmatrix}\text{NUMBER SG}\end{bmatrix} \sqcup \begin{bmatrix}\text{NUMBER SG}\end{bmatrix} = \begin{bmatrix}\text{NUMBER SG}\end{bmatrix}$
- Underspecified $\begin{bmatrix}\text{NUMBER SG}\end{bmatrix} \sqcup \begin{bmatrix}\quad\end{bmatrix} = \begin{bmatrix}\text{NUMBER SG}\end{bmatrix}$
- Different Specs $\begin{bmatrix}\text{NUMBER SG}\end{bmatrix} \sqcup \begin{bmatrix}\text{PERSON 3}\end{bmatrix} = \begin{bmatrix}\text{NUMBER SG}\\ \text{PERSON 3}\end{bmatrix}$
- Conflicting Specs $\begin{bmatrix}\text{NUMBER SG}\end{bmatrix} \sqcup \begin{bmatrix}\text{NUMBER PL}\end{bmatrix} = \varnothing$

Larger Unification Example

$$
\begin{bmatrix} \text{AGREEMENT} & \boxed{1} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \boxed{1} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{PERSON} & 3 \\ \text{NUMBER} & \text{SG} \end{bmatrix} \end{bmatrix} \end{bmatrix} =
$$

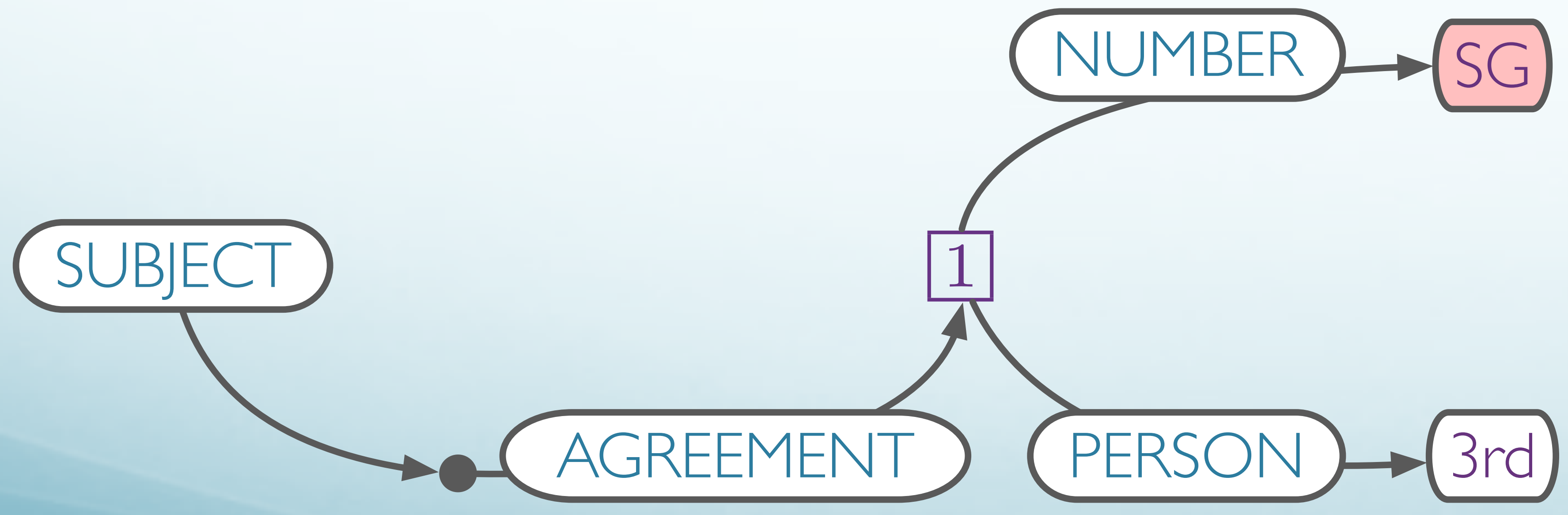
$$
\begin{bmatrix} \text{AGREEMENT} & \boxed{1} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \boxed{1} \begin{bmatrix} \text{PERSON} & 3 \\ \text{NUMBER} & \text{SG} \end{bmatrix} \end{bmatrix} \end{bmatrix}
$$

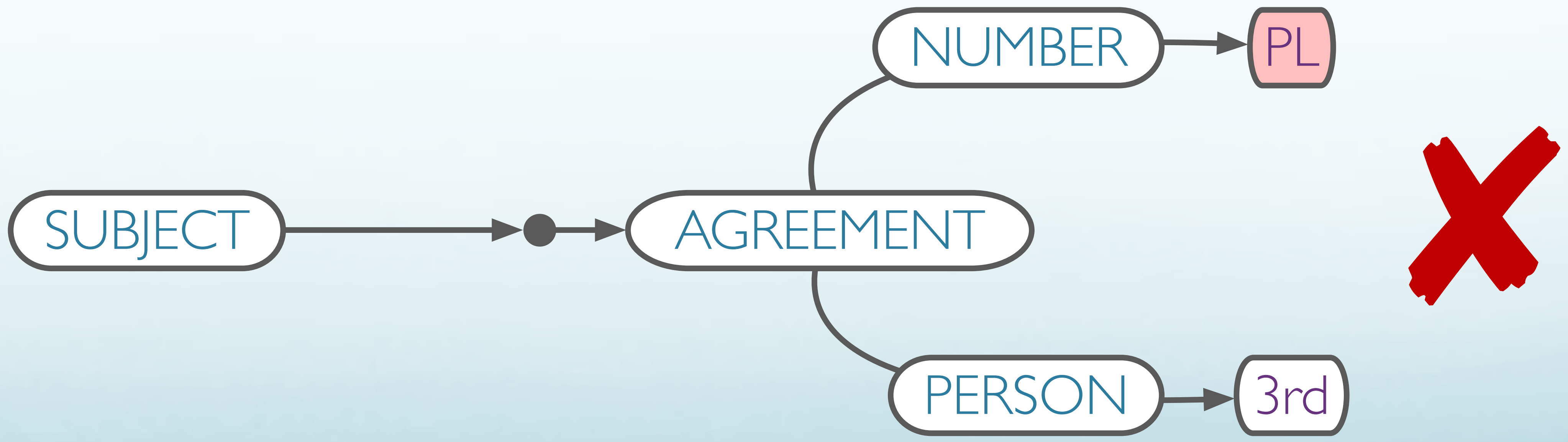
One More Unification Example

$$\begin{bmatrix} \text{AGREEMENT} & \boxed{1}\begin{bmatrix} \text{NUMBER} & sg \\ \text{PERSON} & 3 \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \boxed{1} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & sg \\ \text{PERSON} & 3 \end{bmatrix} \\ \text{SUBJECT} & \begin{bmatrix} \text{AGREEMENT} & \begin{bmatrix} \text{NUMBER} & PL \\ \text{PERSON} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Unification

$$\begin{bmatrix} \text{AGREEMENT } \boxed{1} \begin{bmatrix}\text{NUMBER } sg \\ \text{PERSON } 3\end{bmatrix} \\ \text{SUBJECT } \begin{bmatrix}\text{AGREEMENT } \boxed{1}\end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{AGREEMENT } \begin{bmatrix}\text{NUMBER } sg \\ \text{PERSON } 3\end{bmatrix} \\ \text{SUBJECT } \begin{bmatrix}\text{AGREEMENT } \begin{bmatrix}\text{NUMBER } PL \\ \text{PERSON } 3\end{bmatrix}\end{bmatrix}\end{bmatrix} = \emptyset \quad \text{Failure!}$$



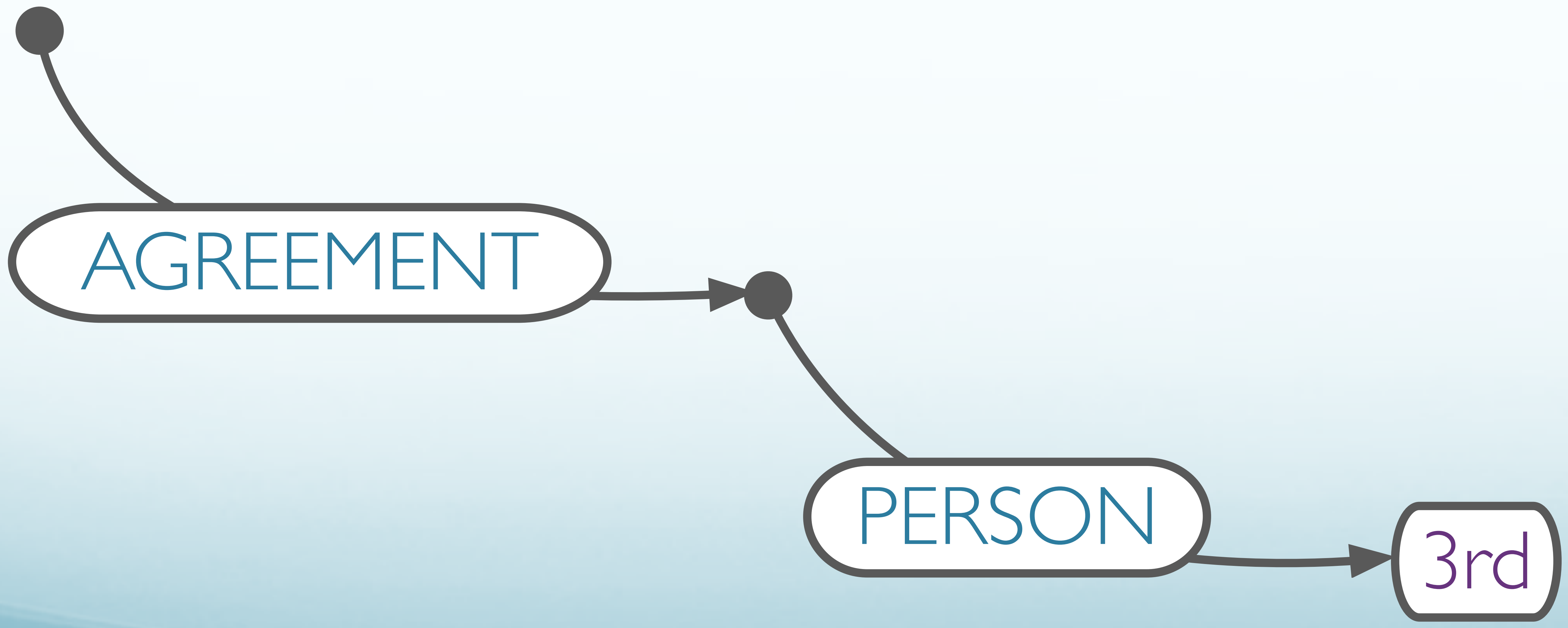


Rule Representation

- $\beta \rightarrow \beta_1 \ldots \beta_n$
  $\{set\ of\ constraints\}$ $\qquad \langle \beta_i\ feature\ path \rangle = Atomic\ value \mid \langle \beta_j\ feature\ path \rangle$
- $Pron \rightarrow$ 'he'

$$\langle \textbf{\textit{P}}\textsc{\textbf{\textit{ron}}}\ \ \text{A}\textsc{greement}\ \ \text{P}\textsc{erson} \rangle = \textbf{\textit{3rd}}$$

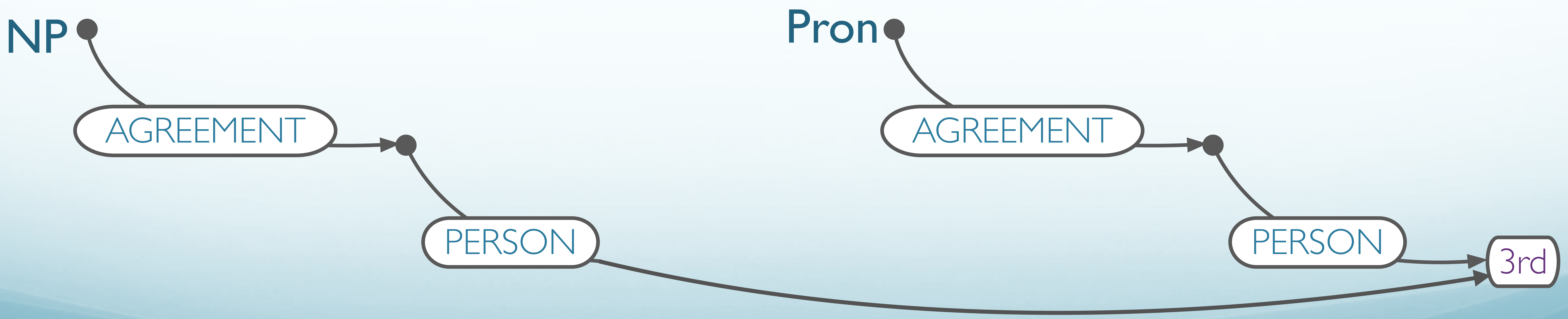
Pron



Rule Representation

- $\beta \rightarrow \beta_1 \ldots \beta_n$
  $\{set\ of\ constraints\}$ $\quad\quad \langle \beta_i\ feature\ path \rangle = Atomic\ value \mid \langle \beta_j\ feature\ path \rangle$
- $NP \rightarrow \textit{Pron}$

$$\langle \boldsymbol{NP}\ \textsc{Agreement Person} \rangle = \langle \boldsymbol{\textsc{Pron}}\ \textsc{Agreement Person} \rangle$$



Agreement with Heads and Features

- $\beta \rightarrow \beta_1 \ldots \beta_n$
  $\{set\ of\ constraints\}$ $\quad \langle \beta_i\ feature\ path \rangle = Atomic\ value \mid \langle \beta_j\ feature\ path \rangle$

$\boldsymbol{S} \rightarrow \boldsymbol{NP}\ \boldsymbol{VP}$
$\langle \boldsymbol{NP}\ \textsc{Agreement} \rangle = \langle \boldsymbol{VP}\ \textsc{Agreement} \rangle$

$\boldsymbol{S} \rightarrow \boldsymbol{Aux}\ \boldsymbol{NP}\ \boldsymbol{VP}$
$\langle \boldsymbol{Aux}\ \textsc{Agreement} \rangle = \langle \boldsymbol{NP}\ \textsc{Agreement} \rangle$

$\boldsymbol{NP} \rightarrow \boldsymbol{Det}\ \boldsymbol{Nominal}$
$\langle \boldsymbol{Det}\ \textsc{Agreement} \rangle = \langle \boldsymbol{Nominal}\ \textsc{Agreement} \rangle$
$\langle \boldsymbol{NP}\ \textsc{Agreement} \rangle = \langle \boldsymbol{Nominal}\ \textsc{Agreement} \rangle$

$\boldsymbol{Aux} \rightarrow does$
$\langle \boldsymbol{Aux}\ \textsc{Agreement Number} \rangle = \boldsymbol{sg}$
$\langle \boldsymbol{NP}\ \textsc{Agreement Person} \rangle = \boldsymbol{3rd}$

$\boldsymbol{Det} \rightarrow this$
$\langle \boldsymbol{Det}\ \textsc{Agreement Number} \rangle = \boldsymbol{sg}$

$\boldsymbol{Det} \rightarrow these$
$\langle \boldsymbol{Det}\ \textsc{Agreement Number} \rangle = \boldsymbol{pl}$

$\boldsymbol{Verb} \rightarrow serve$
$\langle \boldsymbol{Verb}\ \textsc{Agreement Number} \rangle = \boldsymbol{pl}$

$\boldsymbol{Noun} \rightarrow flight$
$\langle \boldsymbol{Noun}\ \textsc{Agreement Number} \rangle = \boldsymbol{sg}$

HW #5

Goals

- Explore the role of features in implementing linguistic constraints.
- Identify some of the challenges in building compact constraints to define a precise grammar.
- Apply feature-based grammars to perform grammar checking.

Tasks

- Build a Feature-Based Grammar
  - We will focus on the building of the grammar itself — you may use NLTK's `nltk.parse.FeatureEarleyChartParser` or similar.

Simple Feature Grammars

- S → NP VP

Simple Feature Grammars

- `S -> NP[NUM=?n] VP[NUM=?n]`
- `NP[NUM=?n] -> N[NUM=?n]`
- `NP[NUM=?n] -> PropN[NUM=?n]`
- `NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]`
- `Det[NUM=sg] -> 'this' | 'every'`
- `Det[NUM=pl] -> 'these' | 'all'`
- `N[NUM=sg] -> 'dog' | 'girl' | 'car' | 'child'`
- `N[NUM=pl] -> 'dogs' | 'girls' | 'cars' | 'children'`

Parsing with Features

```
>>> cp = load_parser('grammars/book_grammars/
feat0.fcfg’)
>>> for tree in cp.parse(tokens):
...     print(tree)

(S[] (NP[NUM='sg']
  (PropN[NUM='sg'] Kim))
    (VP[NUM='sg', TENSE='pres']
      (TV[NUM='sg', TENSE='pres'] likes)
      (NP[NUM='pl'] (N[NUM='pl'] children))))
```

Feature Applications

- Subcategorization
  - Verb-Argument constraints
    - Number, type, characteristics of args
      - e.g. is the subject *animate*?
      - Also adjectives, nouns
- Long-distance dependencies
  - e.g. filler–gap relations in wh-questions

Morphosyntactic Features

- Grammtical feature that influences morphological or syntactic behavior
  - English:
    - Number:
      - Dog, dogs
    - Person:
      - am; are; is
    - Case:
      - I / me; he / him; etc.

Semantic Features

- Grammatical features that influence semantic (meaning) behavior of associated units
- E.g.:
  - *?The rocks slept.*
- Many proposed:
  - `Animacy: +/-`
  - `Gender: masculine, feminine, neuter`
  - `Human: +/-`
  - `Adult: +/-`
  - `Liquid: +/-`

Aspect (J&M 17.4.2)

- *The climber [**hiked**] [**for six hours**].*
- *The climber [**hiked**] [**on Saturday**].*
- *The climber [**reached the summit**] [**on Saturday**].*
- **The climber [**reached the summit**] [**for six hours**].*

- Contrast:
  - *Achievement* (in an instant) vs *activity* (for a time)