The Mathematics of Language

UW Math Hour
March 12, 2017

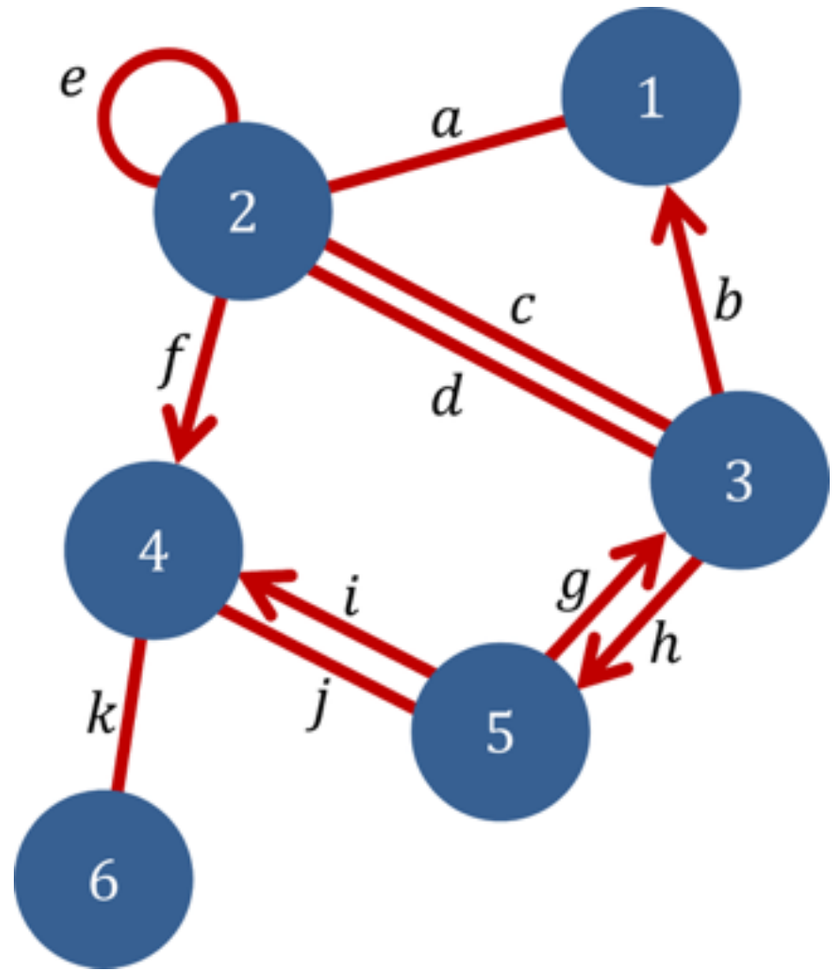
Emily M. Bender
UW Linguistics

The Mathematics of Language (Specifically Syntax)

- Graphs
- Application 1: Trees & tree descriptions
- Modeling grammaticality
- Application 2: Feature structures
- Modeling pizza preferences
- Back to modeling grammaticality
- Disambiguation

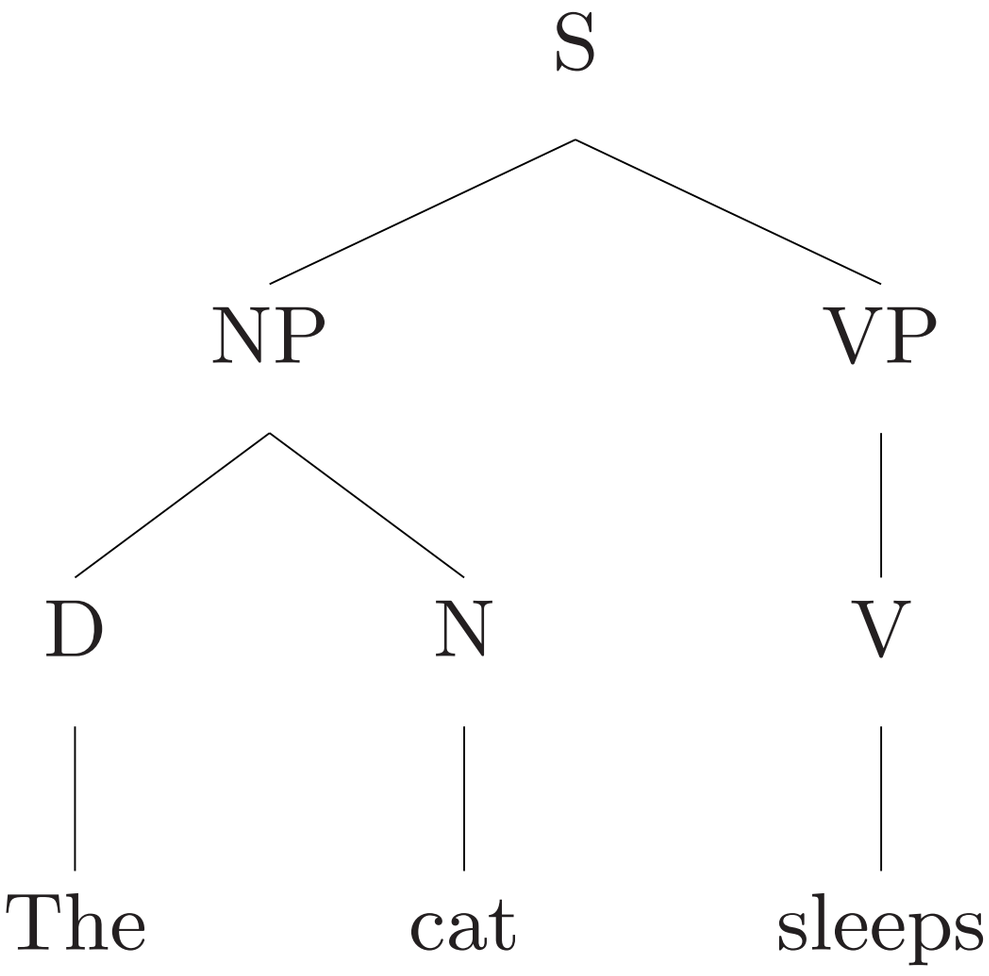
Graphs

- A graph consists of nodes and edges
- Edges connect nodes
- The edges can be directed or undirected
- Edges can be labeled or unlabeled
- Nodes can be labeled or unlabeled

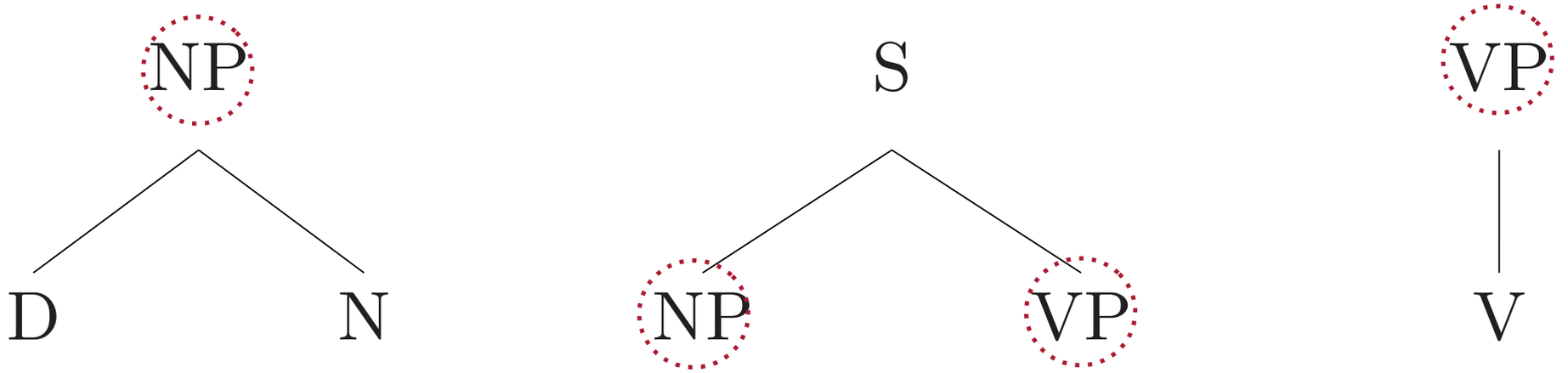


Graphs: application 1
Syntax trees

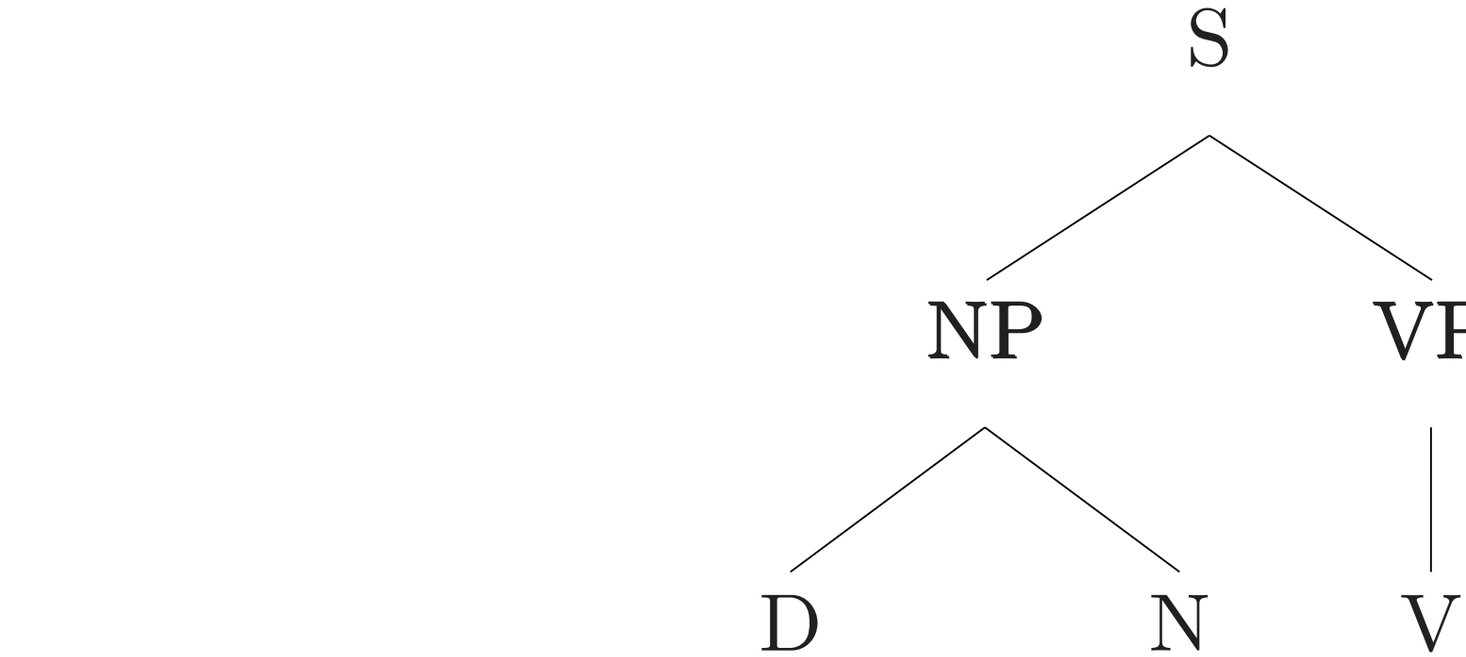
- Trees are a special case of graphs
- Nodes are labeled
- Edges are unlabeled
- Edges are directed
- One node has only out-going arcs
- Nodes have at most one in-coming arc
- Graph is connected
- Graph is acyclic
- Nodes are ordered

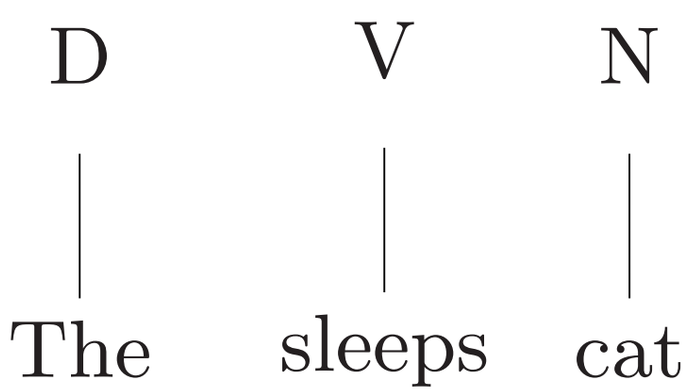


Formal grammar: descriptions of trees

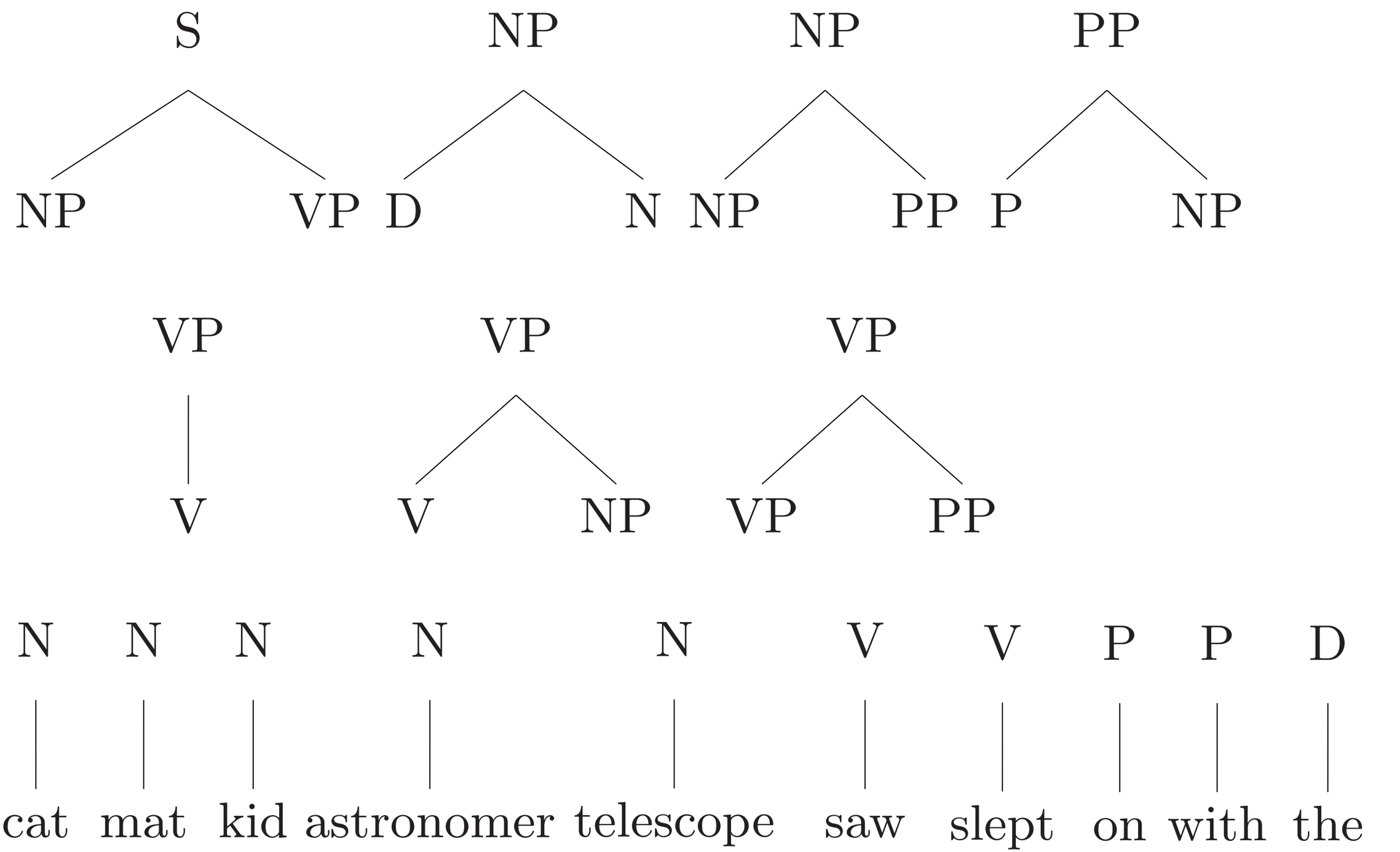


Formal grammar: descriptions of trees





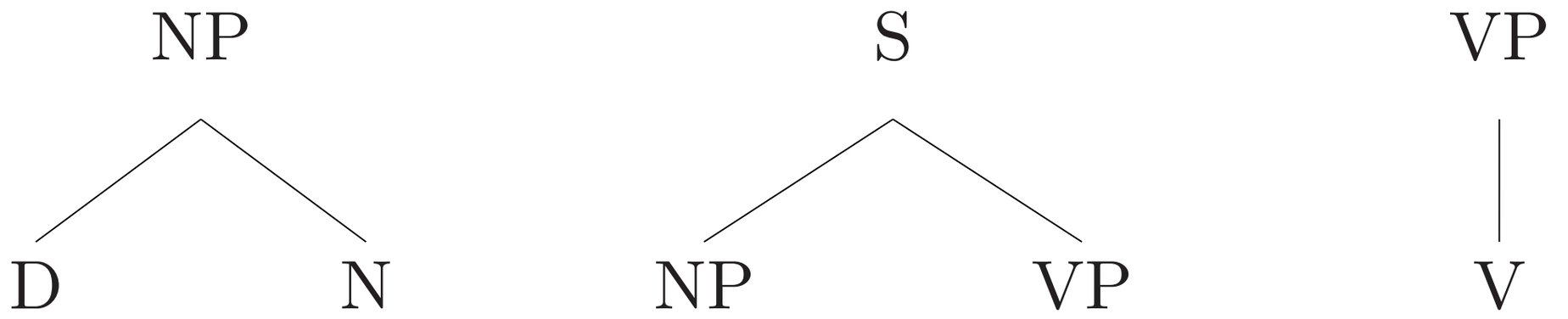
The cat slept on the mat.
The astronomer saw the kid with the telescope.



Modeling grammaticality

- Linguists use formal grammars to model natural languages
- Such a model should distinguish between grammatical and ungrammatical sentences
- And also illuminate how the meanings of words combine to form the meanings of sentences

Treelets as rules

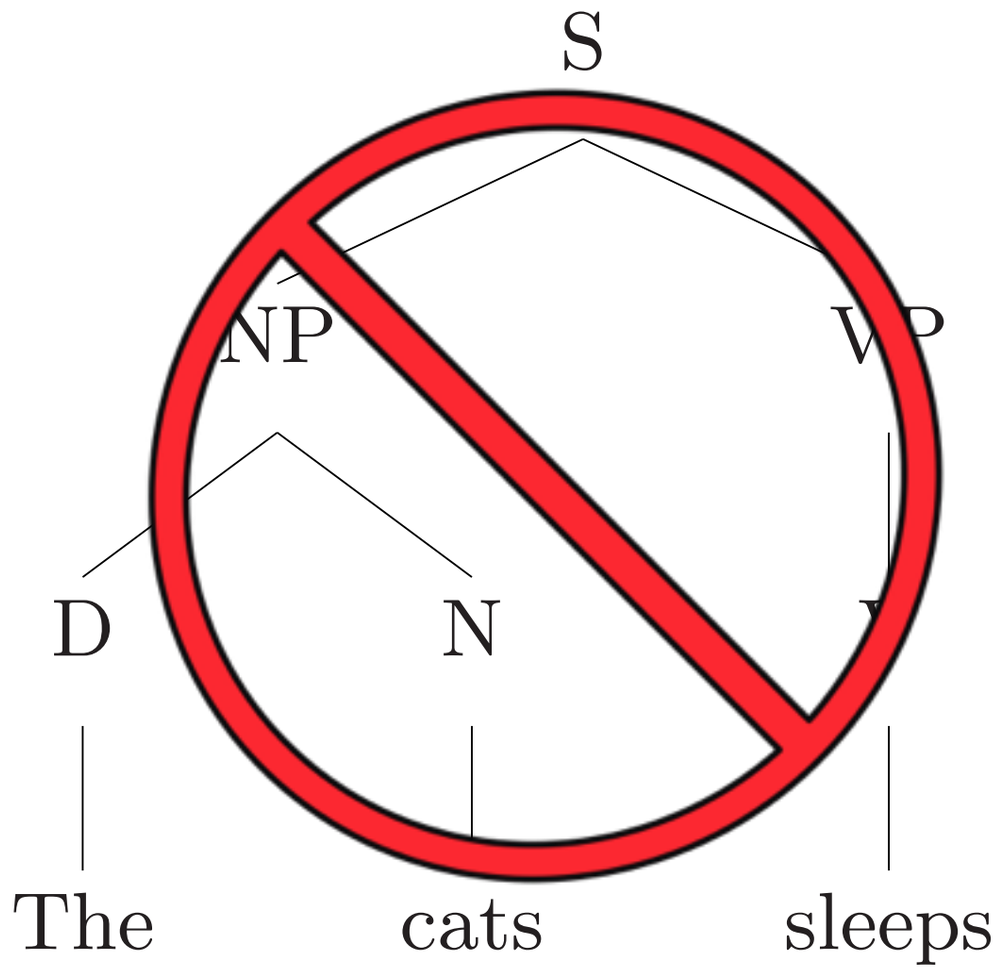


$$
\begin{array}{rcl}
\mathrm{S} & \rightarrow & \mathrm{NP\ VP} \\
\mathrm{NP} & \rightarrow & \mathrm{D\ N} \\
\mathrm{VP} & \rightarrow & \mathrm{V}
\end{array}
$$

More rules

$$
\begin{array}{rcl}
\text{S} & \rightarrow & \text{NP VP} \\
\text{NP} & \rightarrow & \text{D N} \\
\text{NP} & \rightarrow & \text{NP PP} \\
\text{PP} & \rightarrow & \text{P NP} \\
\text{VP} & \rightarrow & \text{V} \\
\text{VP} & \rightarrow & \text{V NP} \\
\text{VP} & \rightarrow & \text{VP PP}
\end{array}
$$

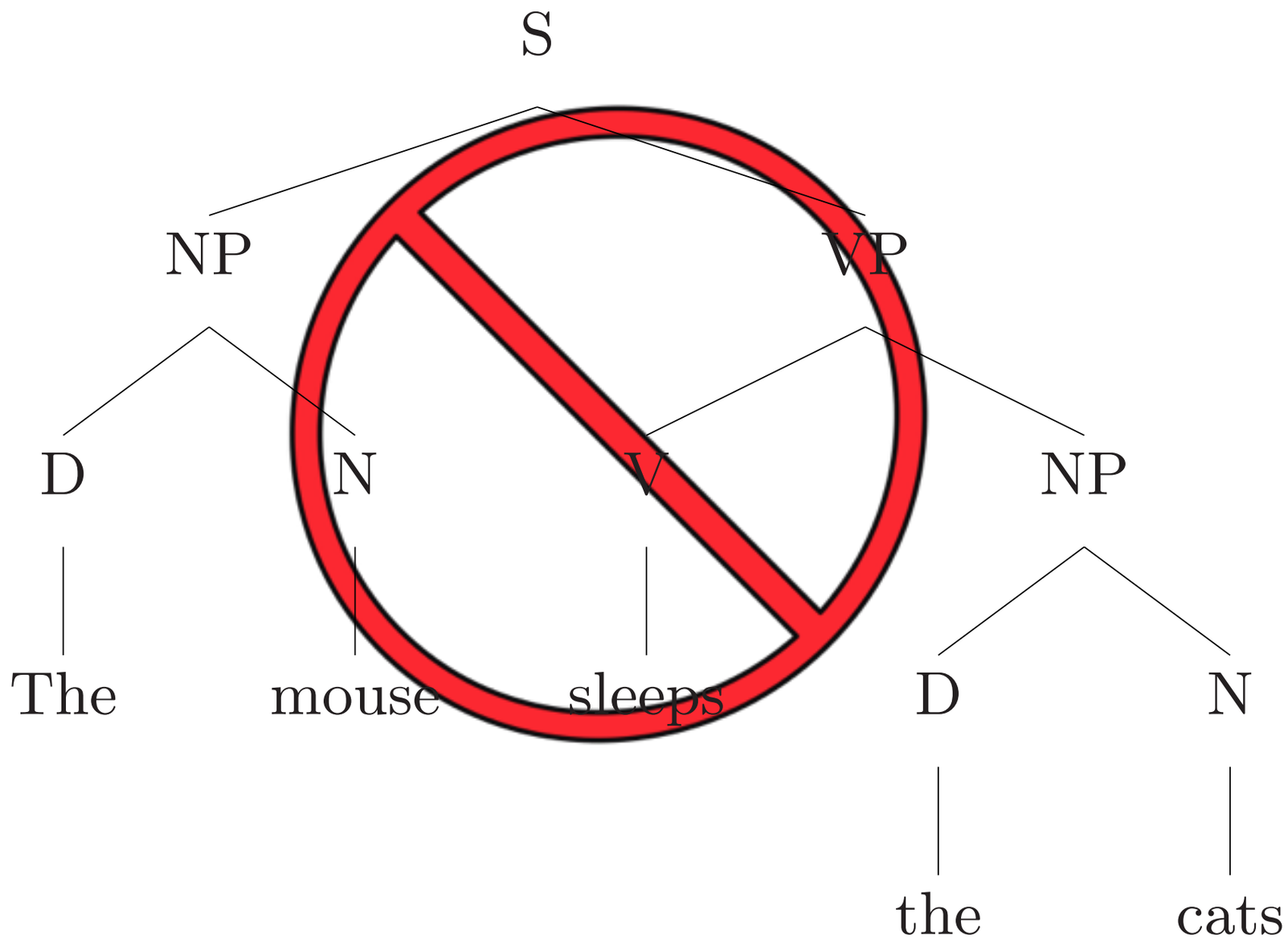
$$
\begin{array}{rcl}
\text{N} & \rightarrow & \text{cats} \\
\text{N} & \rightarrow & \text{mouse} \\
\text{V} & \rightarrow & \text{sleeps} \\
\text{V} & \rightarrow & \text{saw} \\
\text{V} & \rightarrow & \text{give} \\
\text{D} & \rightarrow & \text{the}
\end{array}
$$



More rules

$$
\begin{array}{rcl}
\text{S} & \rightarrow & \text{NP VP} \\
\text{NP} & \rightarrow & \text{D N} \\
\text{NP} & \rightarrow & \text{NP PP} \\
\text{PP} & \rightarrow & \text{P NP} \\
\text{VP} & \rightarrow & \text{V} \\
\text{VP} & \rightarrow & \text{V NP} \\
\text{VP} & \rightarrow & \text{VP PP}
\end{array}
$$

$$
\begin{array}{rcl}
\text{N} & \rightarrow & \text{cats} \\
\text{N} & \rightarrow & \text{mouse} \\
\text{V} & \rightarrow & \text{sleeps} \\
\text{V} & \rightarrow & \text{saw} \\
\text{V} & \rightarrow & \text{give} \\
\text{D} & \rightarrow & \text{the}
\end{array}
$$



Better rules - part I

S	→	NP-pl VP-pl	N-pl	→	cats
S	→	NP-sg VP-sg	N-sg	→	mouse
NP-pl	→	D N-pl	V-sg	→	sleeps
NP-sg	→	D N-sg	V-pl	→	saw
NP-pl	→	NP-pl PP	V-sg	→	saw
NP-sg	→	NP-sg PP	V-pl	→	give
PP	→	P NP-pl	D	→	the
PP	→	P NP-sg			
VP-pl	→	V-pl NP-sg			
VP-pl	→	V-pl NP-pl			
VP-sg	→	V-sg NP-sg			
VP-sg	→	V-sg NP-pl			
VP-pl	→	VP-pl PP			
VP-sg	→	VP-sg PP			

Better rules - part II

S	→	NP VP	N	→	cat
NP	→	D N	N	→	mouse
NP	→	NP PP	V-1	→	slept
PP	→	P NP	V-2	→	saw
VP	→	V-1	V-3	→	gave
VP	→	V-2 NP	V-4	→	put
VP	→	V-3 NP NP	V-5	→	told
VP	→	V-4 NP PP	D	→	the
VP	→	V-5 NP S			
VP	→	VP PP			

Better rules - but too many rules!!

S	→	NP-pl VP-pl
S	→	NP-sg VP-sg
NP-pl	→	D N-pl
NP-sg	→	D N-sg
NP-pl	→	NP-pl PP
NP-sg	→	NP-sg PP
PP	→	P NP-pl
PP	→	P NP-sg
VP-pl	→	V-1-pl
VP-sg	→	V-1-sg
VP-pl	→	V-2-pl NP-sg
VP-pl	→	V-2-pl NP-pl
VP-sg	→	V-2-sg NP-sg
VP-sg	→	V-2-sg NP-pl
VP-pl	→	V-3-pl NP-sg NP-sg
VP-pl	→	V-3-pl NP-pl NP-pl
VP-sg	→	V-3-sg NP-sg NP-sg
VP-sg	→	V-3-sg NP-pl NP-pl
VP-pl	→	V-3-pl NP-sg NP-pl
VP-pl	→	V-3-pl NP-pl NP-sg
VP-sg	→	V-3-sg NP-sg NP-pl
VP-sg	→	V-3-sg NP-pl NP-sg
VP-pl	→	V-2-pl NP-sg PP
VP-pl	→	V-2-pl NP-pl PP
VP-sg	→	V-2-sg NP-sg PP
VP-sg	→	V-2-sg NP-pl PP
VP-pl	→	V-2-pl NP-sg S
VP-pl	→	V-2-pl NP-pl S
VP-sg	→	V-2-sg NP-sg S
VP-sg	→	V-2-sg NP-pl S
VP-pl	→	VP-pl PP
VP-sg	→	VP-sg PP

N-pl	→	cats
N-sg	→	mouse
V-1-sg	→	sleeps
V-1-pl	→	sleep
V-2-sg	→	sees
V-2-pl	→	see
V-3-sg	→	gives
V-3-pl	→	give
V-4-sg	→	puts
V-4-pl	→	put
V-5-sg	→	tells
V-5-pl	→	tell
D	→	the

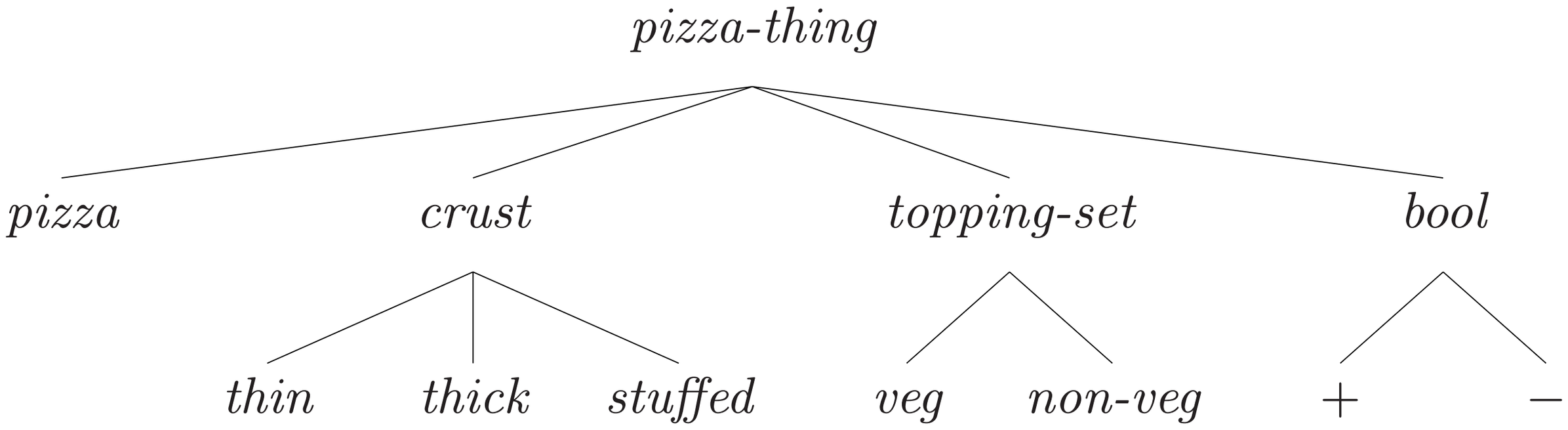
Solution: Feature structures

$$\begin{bmatrix} \textit{phrase} & \\ \text{POS} & \boxed{0} \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} \textit{word} & \\ \text{POS} & \boxed{0} \\ \text{COMPS} & \langle\ \boxed{1},\ \ldots,\ \boxed{n}\ \rangle \end{bmatrix} \boxed{1}\ \ldots\ \boxed{n}$$

$$\text{slept} : \begin{bmatrix} \textit{word} & \\ \text{POS} & \textit{verb} \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \qquad \text{told} : \begin{bmatrix} \textit{word} & \\ \text{POS} & \textit{verb} \\ \text{COMPS} & \langle\ \text{NP},\ \text{S}\ \rangle \end{bmatrix}$$

$$\text{with} : \begin{bmatrix} \textit{word} & \\ \text{POS} & \textit{prep} \\ \text{COMPS} & \langle\ \text{NP}\ \rangle \end{bmatrix}$$

Trees again: Type hierarchy



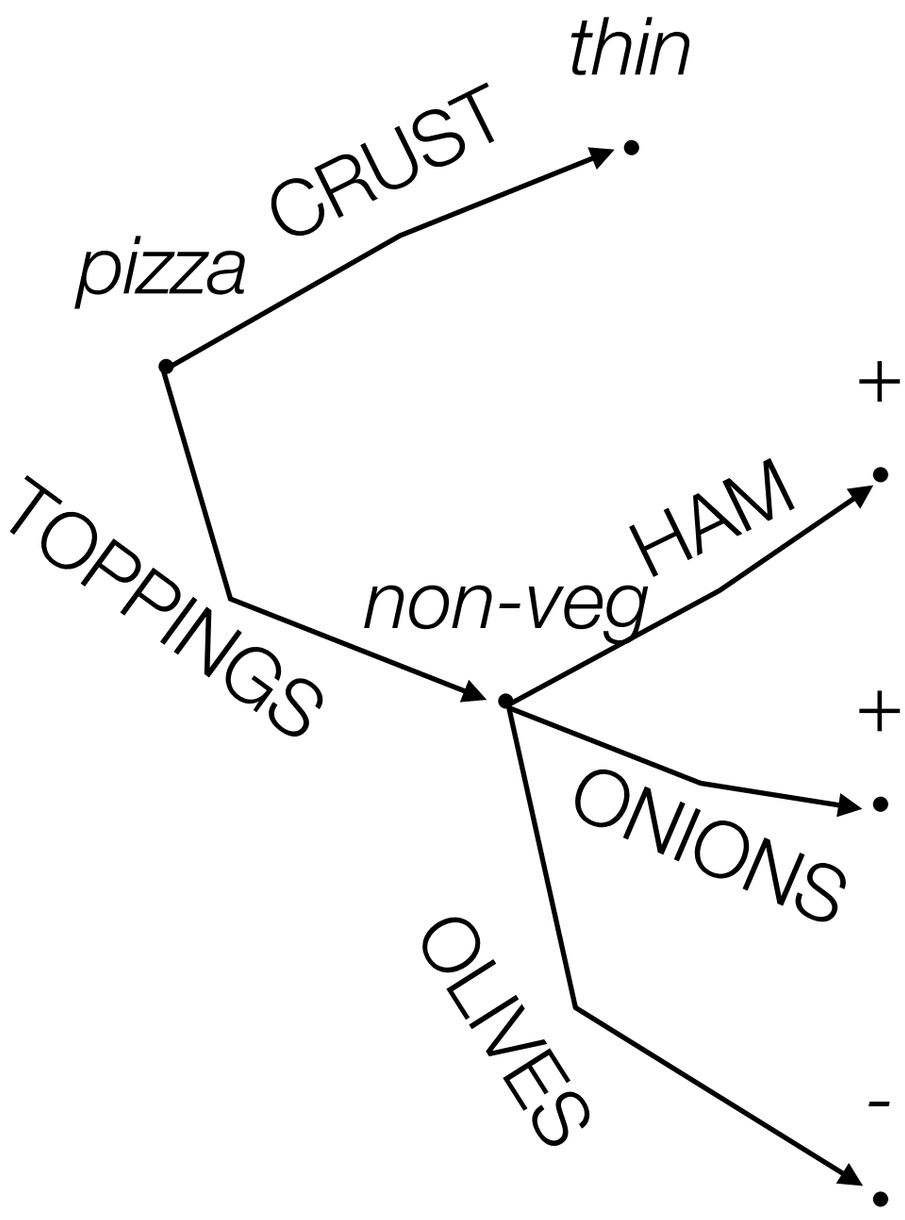
- Trees are a special case of graphs
- Nodes are labeled
- Edges are unlabeled
- Edges are directed
- One node has only out-going arcs

- Nodes have at most one in-coming arc
- Graph is connected
- Graph is acyclic
- ~~Nodes are ordered~~

Type hierarchy: Feature appropriateness

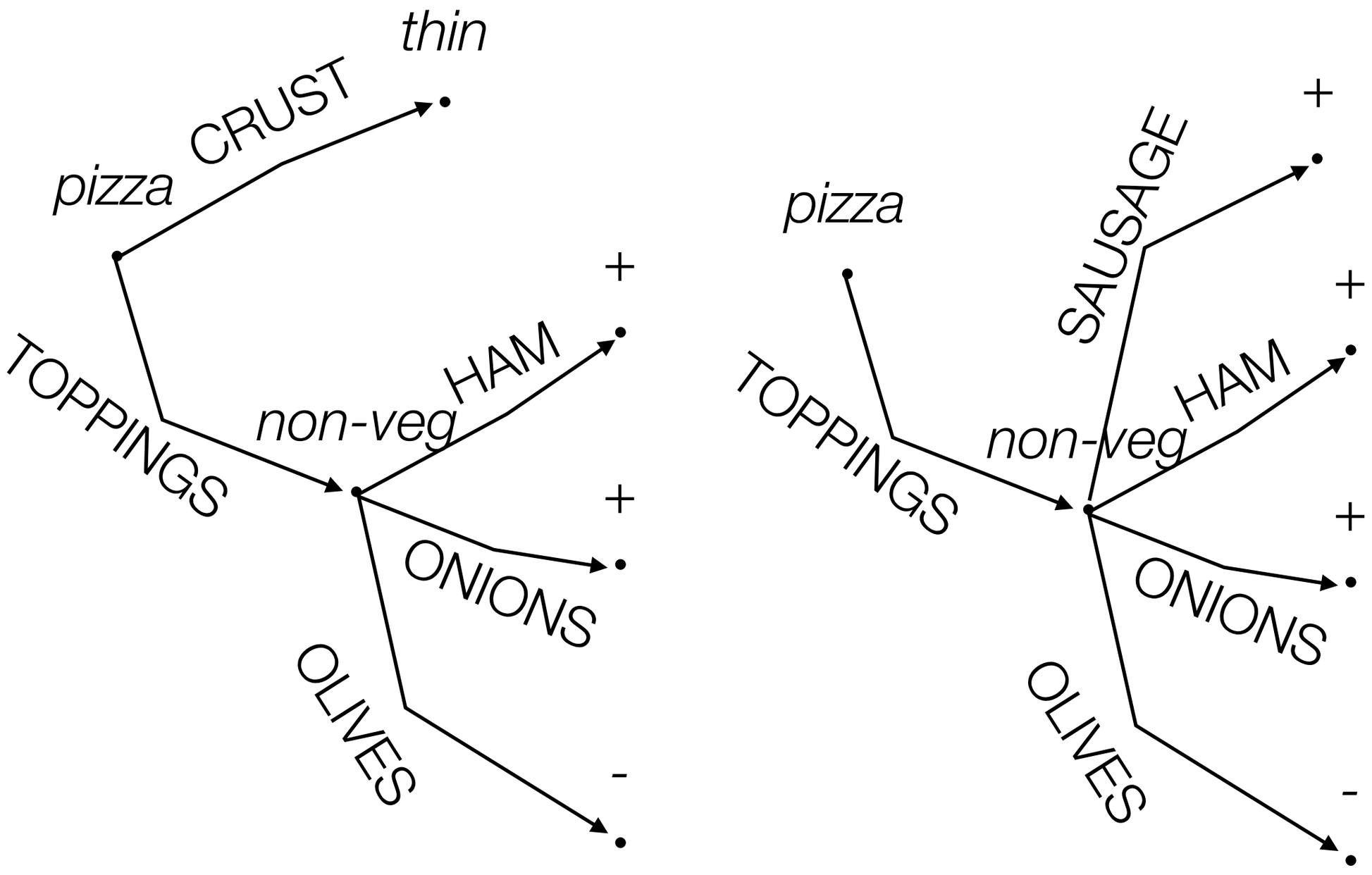
TYPE	FEATURES	IST
pizza-thing		
pizza	$\begin{bmatrix}\text{CRUST} & \textit{crust} \\ \text{TOPPINGS} & \textit{topping-set}\end{bmatrix}$	*pizza-thing*
thin		*crust*
thick		*crust*
stuffed		*crust*
topping-set	$\begin{bmatrix}\text{OLIVES} & \textit{bool} \\ \text{ONIONS} & \textit{bool} \\ \text{MUSHROOMS} & \textit{bool}\end{bmatrix}$	*pizza-thing*
veg		*topping-set*
non-veg	$\begin{bmatrix}\text{SAUSAGE} & \textit{bool} \\ \text{PEPPERONI} & \textit{bool} \\ \text{HAM} & \textit{bool}\end{bmatrix}$	*topping-set*
$+$		*bool*
$-$		*bool*

Graphs: application 2
Typed feature structures



- Nodes are labeled
- Edges are labeled
- Edges are directed
- One node has only out-going arcs
- ~~Nodes have at most one in-coming arc~~
- Graph is connected
- Graph is acyclic
- ~~Nodes are ordered~~

Operation on typed feature structures: Unification



Feature structures in avm notation

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thin} \\ \text{TOPPINGS} & \begin{bmatrix} \textit{non-veg} & \\ \text{HAM} & + \\ \text{ONIONS} & + \\ \text{OLIVES} & - \end{bmatrix} \end{bmatrix}$$

Underspecification

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thin} \\ \text{TOPPINGS} & \begin{bmatrix} \textit{veg} & \\ \text{ONIONS} & + \\ \text{OLIVES} & + \end{bmatrix} \end{bmatrix}$$

- How many fully specified pizza descriptions does this underspecified one correspond to?

Underspecification

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thin} \\ \text{TOPPINGS} & \begin{bmatrix} \textit{veg} & \\ \text{ONIONS} & + \\ \text{OLIVES} & + \end{bmatrix} \end{bmatrix}$$

- How many pizzas in the world does this pizza description correspond to?
- ‘Type/token’ distinction — also applies to sentences!

Combining constraints: Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} & \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \end{bmatrix} \end{bmatrix}$$

$$=$$

$$\begin{bmatrix} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{ONIONS} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix}$$

Combining constraints: Unification

$$\left[\begin{array}{ll} \textit{pizza} & \\ \text{CRUST} & \textit{thick} \\ \text{TOPPINGS} & \left[\begin{array}{ll} \text{OLIVES} & + \\ \text{HAM} & - \end{array}\right] \end{array}\right] \;\&\; \left[\begin{array}{ll} \textit{pizza} & \\ \text{CRUST} & \textit{thin} \\ \text{TOPPINGS} & \left[\begin{array}{ll} \text{OLIVES} & + \\ \text{ONIONS} & + \end{array}\right] \end{array}\right]$$

Combining constraints: Unification

$$\begin{bmatrix} pizza \\ \text{CRUST} & thick \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & + \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} pizza \\ \text{CRUST} & thick \\ \text{TOPPINGS} & veg \end{bmatrix}$$

Combining constraints: Unification

$$\begin{bmatrix} pizza \\ \text{CRUST} & thick \\ \text{TOPPINGS} & \begin{bmatrix} \text{OLIVES} & + \\ \text{HAM} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} pizza \\ \text{CRUST} & thick \\ \text{TOPPINGS} & veg \end{bmatrix}$$

A new theory of pizzas

TYPE	FEATURES	IST
pizza-thing		
pizza	$\begin{bmatrix} \text{CRUST} & \textit{crust} \\ \text{ONE-HALF} & \textit{topping-set} \\ \text{OTHER-HALF} & \textit{topping-set} \end{bmatrix}$	*pizza-thing*
thin		*crust*
thick		*crust*
stuffed		*crust*
topping-set	$\begin{bmatrix} \text{OLIVES} & \textit{bool} \\ \text{ONIONS} & \textit{bool} \\ \text{MUSHROOMS} & \textit{bool} \end{bmatrix}$	*pizza-thing*
veg		*topping-set*
non-veg	$\begin{bmatrix} \text{SAUSAGE} & \textit{bool} \\ \text{PEPPERONI} & \textit{bool} \\ \text{HAM} & \textit{bool} \end{bmatrix}$	*topping-set*
+		*bool*
−		*bool*

Combining constraints: Unification

$$\begin{bmatrix} pizza \\ \text{ONE-HALF} & \begin{bmatrix} \text{ONIONS} & + \\ \text{OLIVES} & - \end{bmatrix} \end{bmatrix} \;\&\; \begin{bmatrix} pizza \\ \text{OTHER-HALF} & \begin{bmatrix} \text{ONIONS} & - \\ \text{OLIVES} & + \end{bmatrix} \end{bmatrix}$$

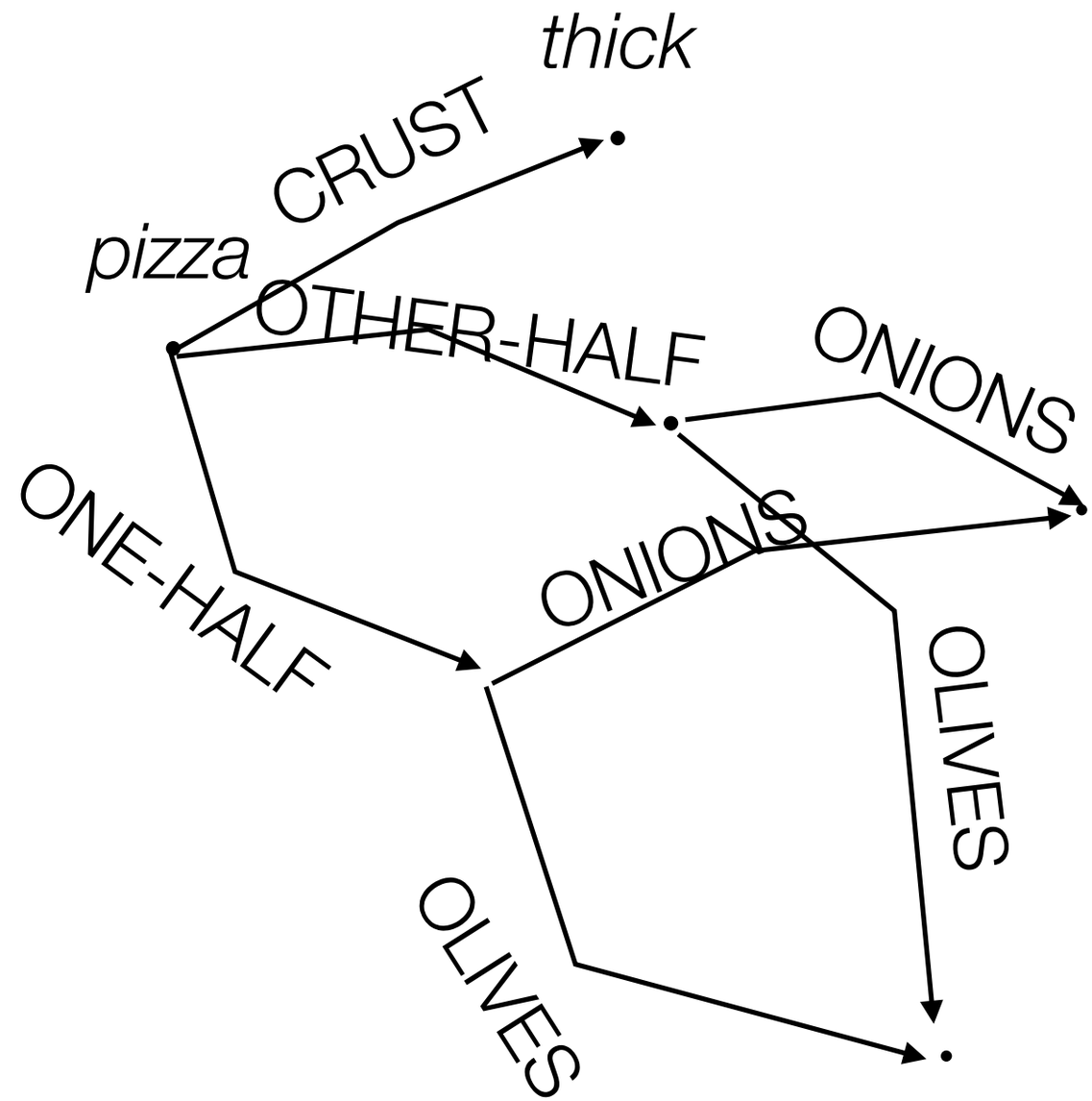
$$=$$

$$\begin{bmatrix} pizza \\ \text{ONE-HALF} & \begin{bmatrix} \text{ONIONS} & + \\ \text{OLIVES} & - \end{bmatrix} \\ \text{OTHER-HALF} & \begin{bmatrix} \text{ONIONS} & - \\ \text{OLIVES} & + \end{bmatrix} \end{bmatrix}$$

Identity constraints (tags)

$$
\begin{bmatrix}
\textit{pizza} & \\
\text{CRUST} & \textit{thick} \\
\text{ONE-HALF} & \begin{bmatrix} \text{OLIVES} & \boxed{1} \\ \text{ONIONS} & \boxed{2} \end{bmatrix} \\
\text{OTHER-HALF} & \begin{bmatrix} \text{OLIVES} & \boxed{1} \\ \text{ONIONS} & \boxed{2} \end{bmatrix}
\end{bmatrix}
$$

Identity constraints (tags)



Combining constraints: Unification

$$\begin{bmatrix} \textit{pizza} & \\ \text{ONE-HALF} & \boxed{1}\begin{bmatrix}\text{ONIONS} & + \\ \text{OLIVES} & -\end{bmatrix} \\ \text{OTHER-HALF} & \boxed{1} \end{bmatrix} \;\&\; \begin{bmatrix} \textit{pizza} & \\ \text{OTHER-HALF} & \begin{bmatrix}\text{MUSHROOMS} & - \\ \text{OLIVES} & -\end{bmatrix} \end{bmatrix}$$

$$=$$

$$\begin{bmatrix} \textit{pizza} & \\ \text{ONE-HALF} & \boxed{1}\begin{bmatrix}\text{ONIONS} & + \\ \text{OLIVES} & - \\ \text{MUSHROOMS} & -\end{bmatrix} \\ \text{OTHER-HALF} & \boxed{1} \end{bmatrix}$$

Combining constraints: Unification

$$\begin{bmatrix} pizza \\ \text{ONE-HALF} & \boxed{1}\begin{bmatrix}\text{OLIVES} & + \\ \text{ONIONS} & +\end{bmatrix} \\ \text{OTHER-HALF} & \boxed{1} veg \end{bmatrix} \;\&\; \begin{bmatrix} pizza \\ \text{ONE-HALF} & \begin{bmatrix}\text{SAUSAGE} & + \\ \text{HAM} & -\end{bmatrix} \end{bmatrix}$$

Your turn: Assume this is the menu
Write down your pizza constraints

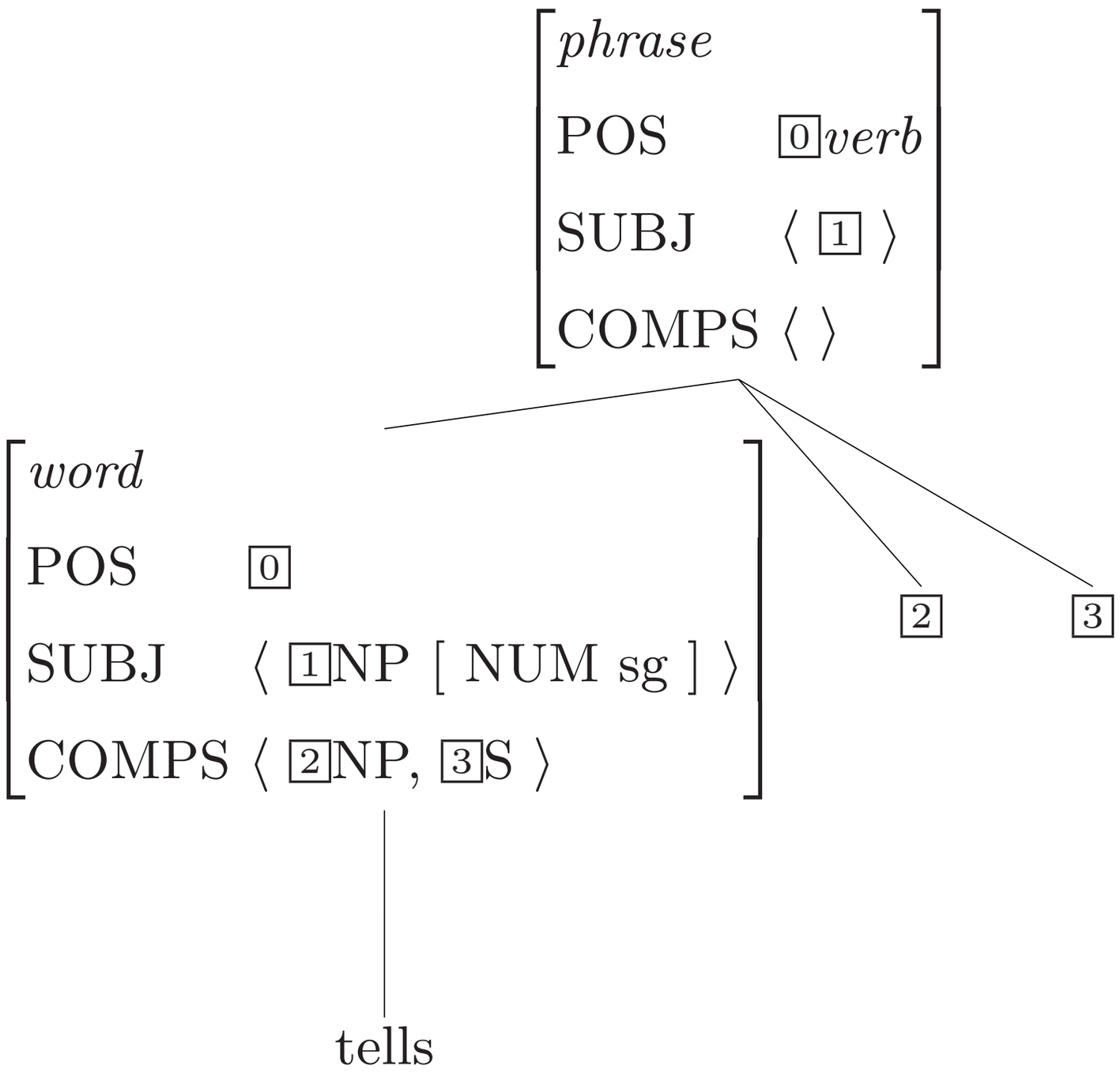
TYPE	FEATURES	IST
pizza-thing		
pizza	$\begin{bmatrix} \text{CRUST} & \textit{crust} \\ \text{ONE-HALF} & \textit{topping-set} \\ \text{OTHER-HALF} & \textit{topping-set} \end{bmatrix}$	*pizza-thing*
thin		*crust*
thick		*crust*
stuffed		*crust*
topping-set	$\begin{bmatrix} \text{OLIVES} & \textit{bool} \\ \text{ONIONS} & \textit{bool} \\ \text{MUSHROOMS} & \textit{bool} \end{bmatrix}$	*pizza-thing*
veg		*topping-set*
non-veg	$\begin{bmatrix} \text{SAUSAGE} & \textit{bool} \\ \text{PEPPERONI} & \textit{bool} \\ \text{HAM} & \textit{bool} \end{bmatrix}$	*topping-set*
+		*bool*
−		*bool*

Unification and modeling language

$$\begin{bmatrix} phrase & \\ \text{POS} & \boxed{0} \\ \text{SUBJ} & \boxed{1} \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \rightarrow \mathbf{H} \begin{bmatrix} word & \\ \text{POS} & \boxed{0} \\ \text{SUBJ} & \boxed{1} \\ \text{COMPS} & \langle\ \boxed{2},\ \ldots,\ \boxed{n}\ \rangle \end{bmatrix} \boxed{2}\ \ldots\ \boxed{n}$$

$$\text{tells} : \begin{bmatrix} word & \\ \text{POS} & verb \\ \text{SUBJ} & \langle\ \text{NP}\ [\ \text{NUM sg}\ ]\ \rangle \\ \text{COMPS} & \langle\ \text{NP, S}\ \rangle \end{bmatrix}$$

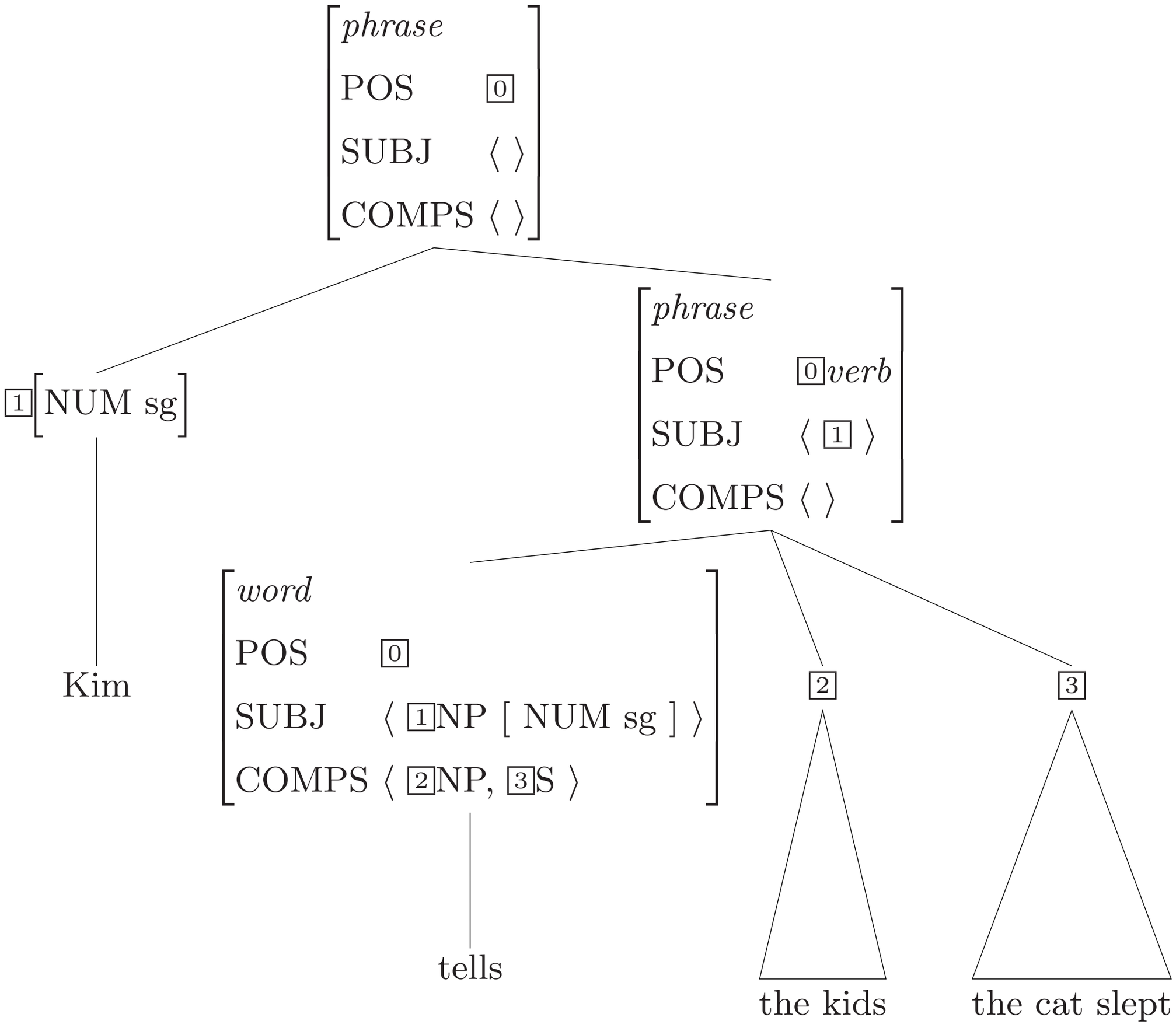
Unification and modeling language



Unification and modeling language

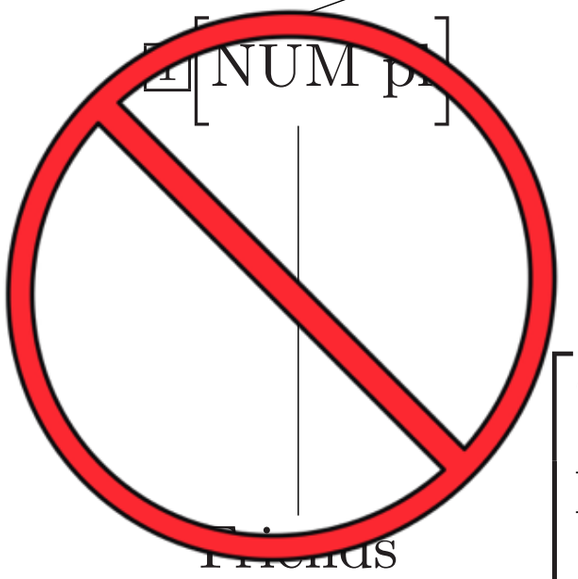
$$\begin{bmatrix} \textit{phrase} & \\ \text{POS} & \boxed{0} \\ \text{SUBJ} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix} \rightarrow \boxed{1}\ \mathbf{H} \begin{bmatrix} \text{POS} & \boxed{0} \\ \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}$$

$$\text{tells} : \begin{bmatrix} \textit{word} & \\ \text{POS} & \textit{verb} \\ \text{SUBJ} & \langle\ \text{NP}\ [\ \text{NUM sg}\ ]\ \rangle \\ \text{COMPS} & \langle\ \text{NP, S}\ \rangle \end{bmatrix}$$


phrase
POS 0
SUBJ ⟨ ⟩
COMPS ⟨ ⟩
1 NUM sg
phrase
POS 0 verb
SUBJ ⟨ 1 ⟩
COMPS ⟨ ⟩
Kim
word
POS 0
SUBJ ⟨ 1 NP [NUM sg] ⟩
COMPS ⟨ 2 NP, 3 S ⟩
tells
2
3
the kids
the cat slept

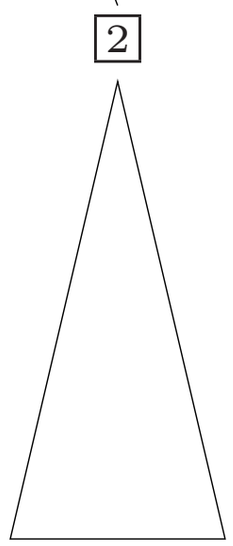
$$\begin{bmatrix} phrase & \\ \text{POS} & \boxed{0} \\ \text{SUBJ} & \langle\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}$$

$$\begin{bmatrix} phrase & \\ \text{POS} & \boxed{0} verb \\ \text{SUBJ} & \langle\ \boxed{1}\ \rangle \\ \text{COMPS} & \langle\ \rangle \end{bmatrix}$$

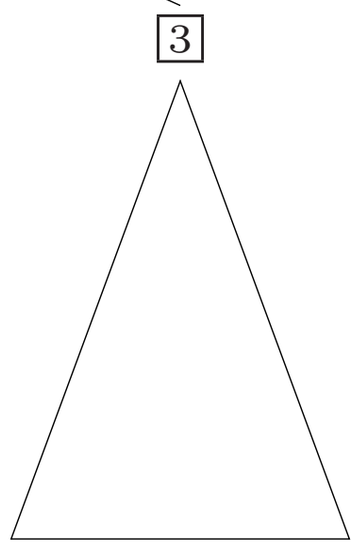


$$\begin{bmatrix} word & \\ \text{POS} & \boxed{0} \\ \text{SUBJ} & \langle\ \boxed{1}\text{NP}\ [\ \text{NUM sg}\ ]\ \rangle \\ \text{COMPS} & \langle\ \boxed{2}\text{NP},\ \boxed{3}\text{S}\ \rangle \end{bmatrix}$$

tells



the kids

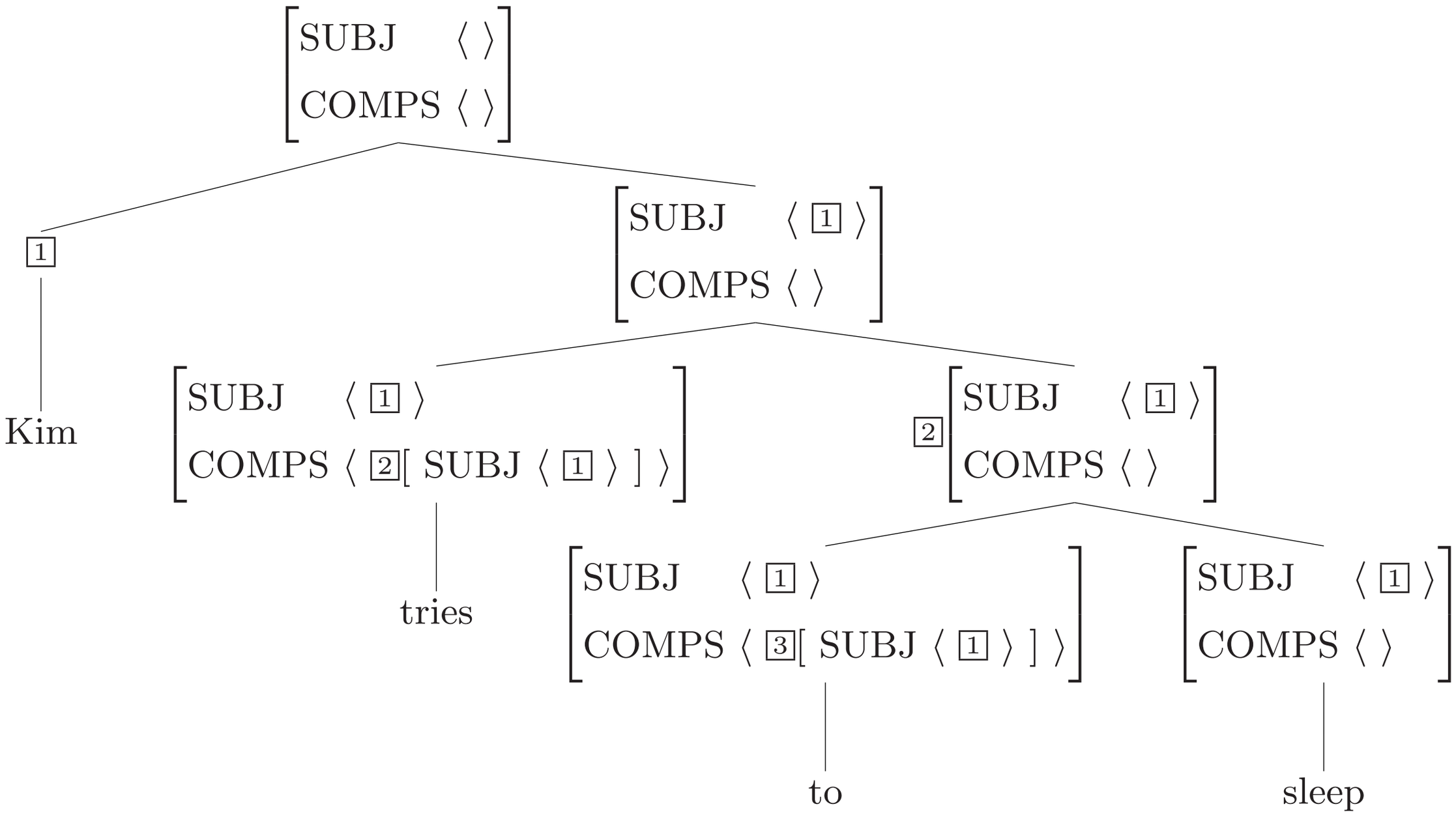


the cat slept

Unification and modeling language

$$\text{tries}: \begin{bmatrix} \textit{word} & \\ \text{POS} & \textit{verb} \\ \text{SUBJ} & \langle\ \boxed{1}\text{NP}\ [\ \text{NUM sg}\ ]\ \rangle \\ \text{COMPS} & \langle\ \text{VP}\ [\ \text{SUBJ}\ \langle\ \boxed{1}\ \rangle\ ]\ \rangle \end{bmatrix}$$

$$\text{to}: \begin{bmatrix} \textit{word} & \\ \text{POS} & \textit{verb} \\ \text{SUBJ} & \langle\ \boxed{1}\text{NP}\ \rangle \\ \text{COMPS} & \langle\ \text{VP}\ [\ \text{SUBJ}\ \langle\ \boxed{1}\ \rangle\ ]\ \rangle \end{bmatrix}$$


SUBJ ⟨ ⟩
COMPS ⟨ ⟩
1
SUBJ ⟨ 1 ⟩
COMPS ⟨ ⟩
Kim
SUBJ ⟨ 1 ⟩
COMPS ⟨ 2[SUBJ ⟨ 1 ⟩] ⟩
2
SUBJ ⟨ 1 ⟩
COMPS ⟨ ⟩
tries
SUBJ ⟨ 1 ⟩
COMPS ⟨ 3[SUBJ ⟨ 1 ⟩] ⟩
SUBJ ⟨ 1 ⟩
COMPS ⟨ ⟩
to
sleep

Scaling up

- Develop algorithms for parsing
- Develop machine-readable formalism for grammar rules
- Write grammar rules
  - English Resource Grammar (erg.delph-in.net)
  - Grammar Matrix (www.delph-in.net/matrix)
- Build treebanks & train statistical models for disambiguation

Disambiguation

- Humans barely notice ambiguity
    - Time flies like an arrow
    - Fruit flies like a banana
- We apply world knowledge to filter out readings that don’t make sense
- Machines instead rely on models of frequent/likely combinations

The Mathematics of Language (Specifically Syntax)

- Graphs
- Application 1: Trees & tree descriptions
- Modeling grammaticality
- Application 2: Feature structures
- Modeling pizza preferences
- Back to modeling grammaticality
- Disambiguation

To learn more

- NACLO: The North American Computational Linguistics Olympiad (nacloweb.org)

- LSA: The Linguistic Society of America (www.linguisticsociety.org)

- Linguistics blogs: languagelog.ldc.upenn.edu, allthingslinguistic.com

- CLMS on facebook: www.facebook.com/uwclma/