Feature-based Parsing
+
Computational Semantics

LING 571 — Deep Processing for NLP
October 28, 2020
Shane Steinert-Threlkeld

Announcements

- HW4:
  - No improvements (e.g. upper/lower-case) in first 3 parts of assignment
    - Parser will miss some sentences :)
  - In shell script for part 5:
    - Hard code **full** paths to `evalb` and `parses.gold` in part 5 of assignment

Ambiguity of the Week

Adam Macqueen
@adam_macqueen

Personally feel not enough hospitals are named after sandwiches.


```
(ROOT
  (S
    (NP (NNP Extinction) (NNP Rebellion) (NNP protester))
    (VP (VBD dressed)
      (SBAR (IN as)
        (S
          (NP (NNP Boris) (NNP Johnson))
          (VP (VBZ scales)
            (NP (NNP Big) (NNP Ben))))))
    (. .)))
```

Ambiguity of the Week

Adam Macqueen
@adam_macqueen

Personally feel not enough hospitals are named after sandwiches.

https://www.theguardian.com/environment/video/2019/oct/18/extinction-rebellion-protester-dressed-as-boris-johnson-scales-big-ben-video

```
(ROOT
  (S
    (NP (NNP Extinction) (NNP Rebellion) (NNP protester))
    (VP (VBD dressed)
      (SBAR (IN as)
        (S
          (NP (NNP Boris) (NNP Johnson))
          (VP (VBZ scales)
            (NP (NNP Big) (NNP Ben))))))
    (. .)))
```

Ambiguity of the Week

Adam Macqueen
@adam_macqueen

Personally feel not enough hospitals are named after sandwiches.


```
(ROOT
  (S
    (NP (NNS Hospitals))
    (VP (VBD named)
      (SBAR (IN after)
        (S
          (NP (NNS sandwiches))
          (VP (VBP kill)
            (NP (CD five))))))
    (. .)))
```

https://www.theguardian.com/environment/video/2019/oct/18/extinction-rebellion-protester-dressed-as-boris-johnson-scales-big-ben-video

```
(ROOT
  (S
    (NP (NNP Extinction) (NNP Rebellion) (NNP protester))
    (VP (VBD dressed)
      (SBAR (IN as)
        (S
          (NP (NNP Boris) (NNP Johnson))
          (VP (VBZ scales)
            (NP (NNP Big) (NNP Ben))))))
    (. .)))
```

http://nlp.stanford.edu:8080/parser/index.jsp

Roadmap

- Feature-based parsing
- Computational Semantics
  - Introduction
  - Semantics
  - Representing Meaning
    - First-Order Logic
    - Events
- HW#5
  - Feature grammars in NLTK
  - Practice with animacy

Computational Semantics

Dialogue System

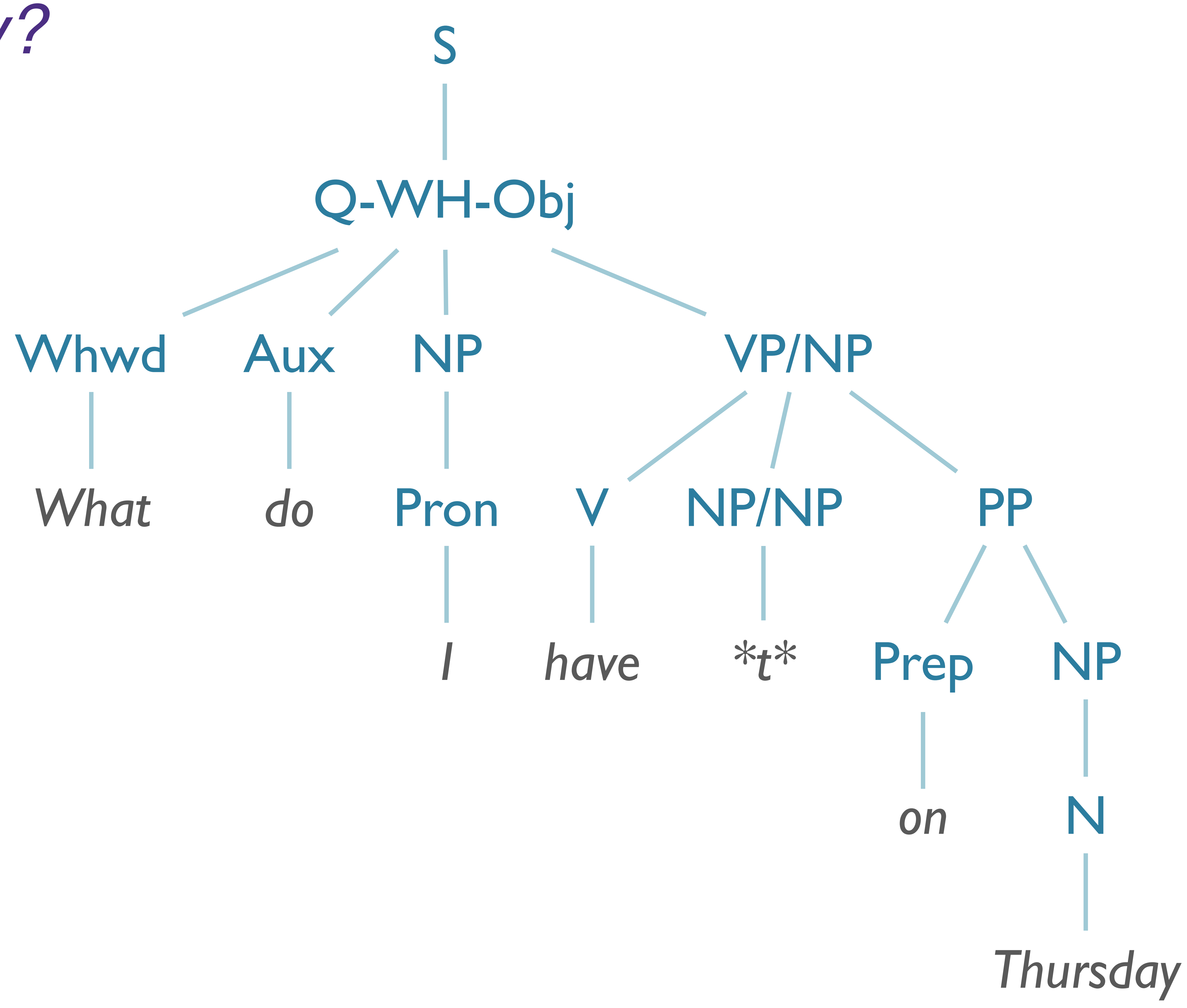
- User: *What do I have on Thursday?*

Dialogue System

- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!

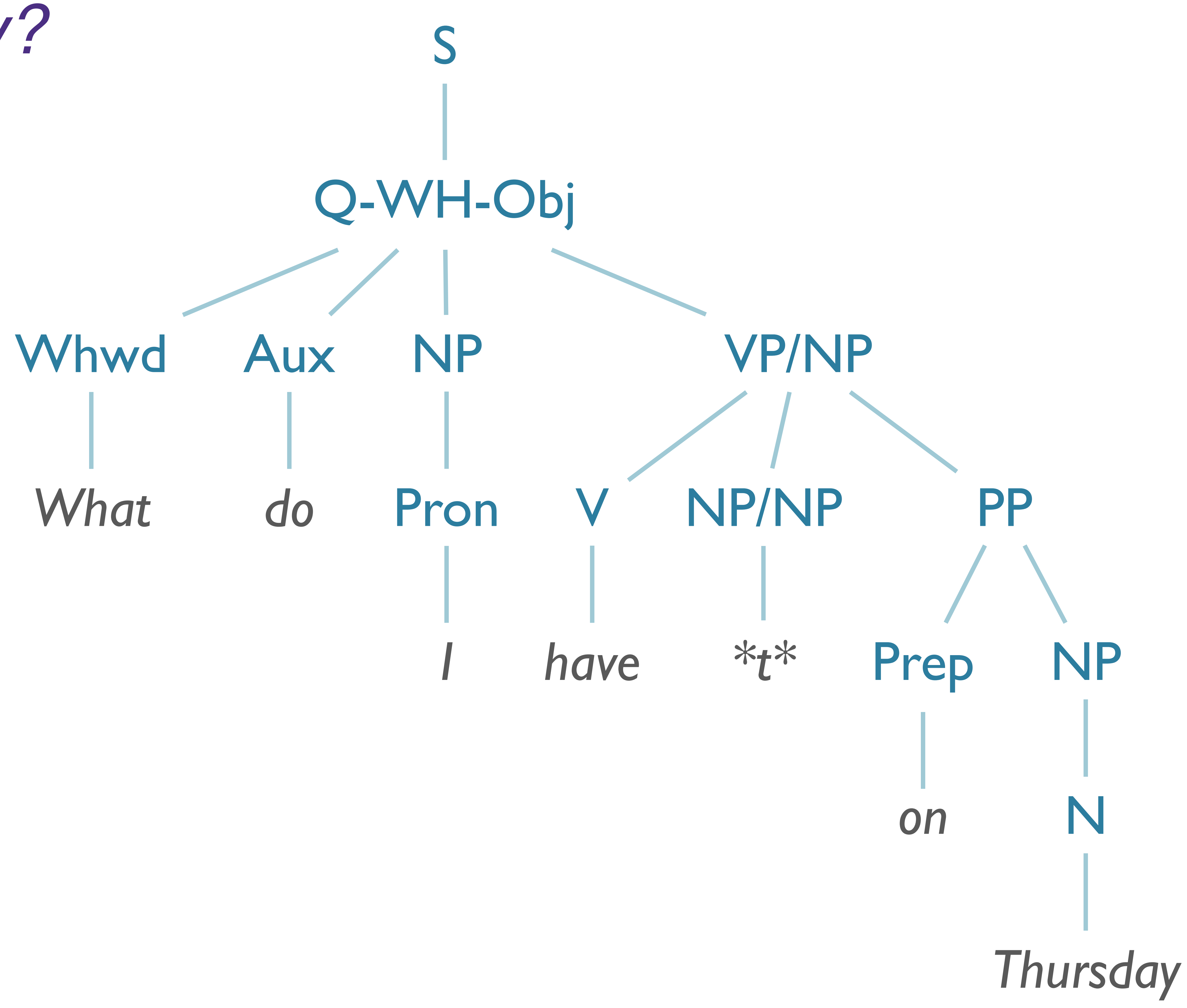
Dialogue System

- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!
  - Here's the structure!



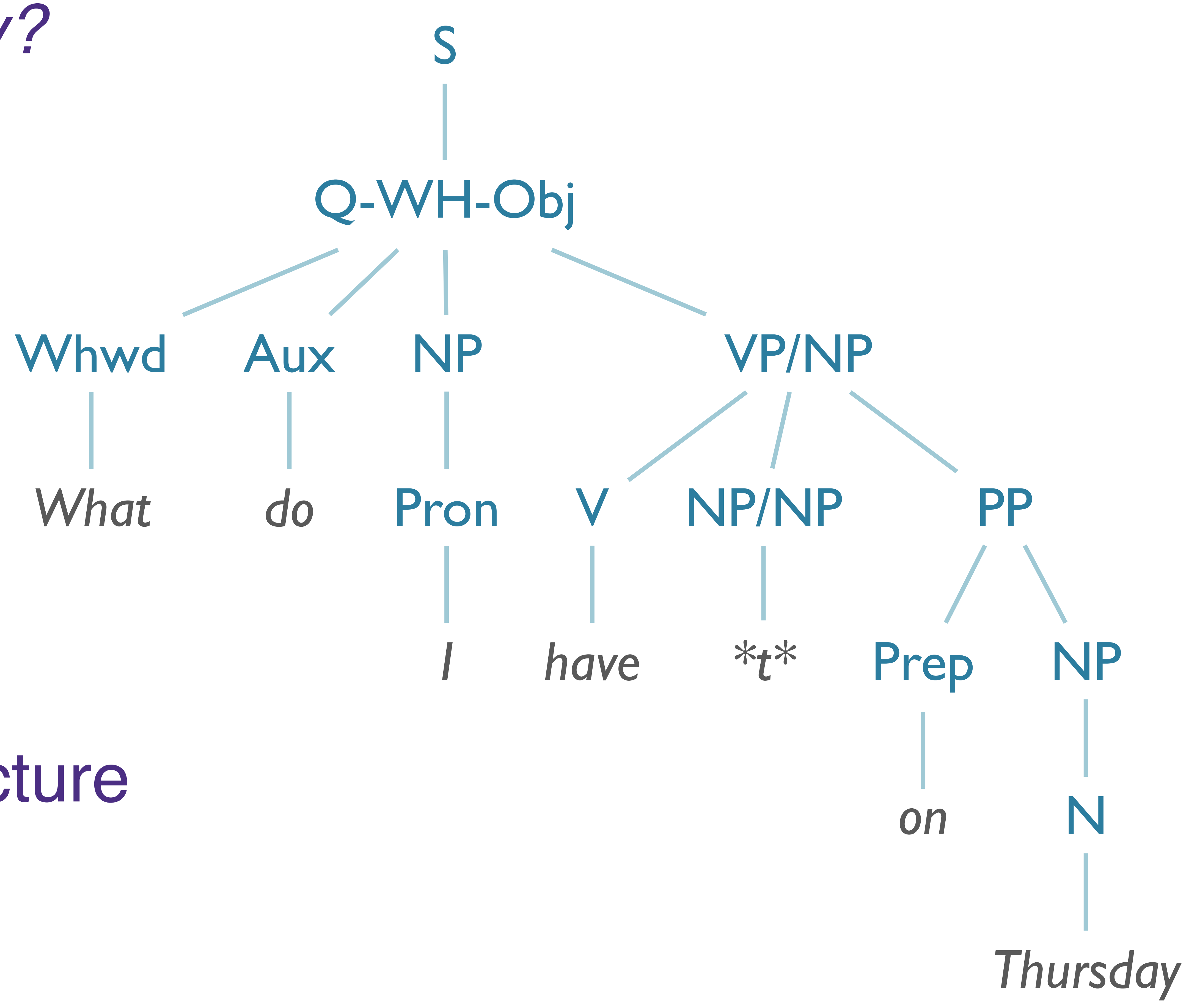
Dialogue System

- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!
  - Here's the structure!
- System:
  - Great, but what do I *DO* now?

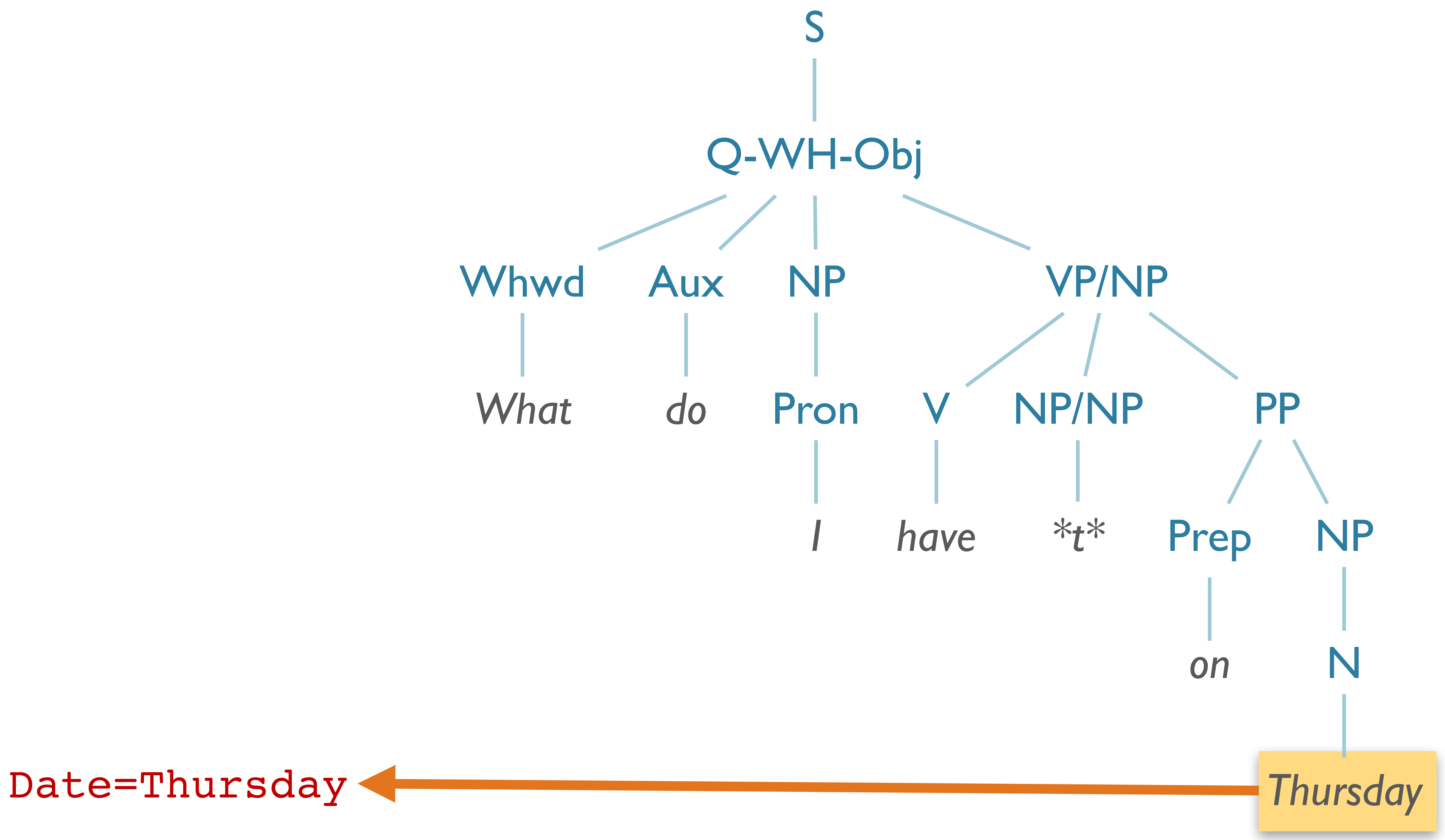


Dialogue System

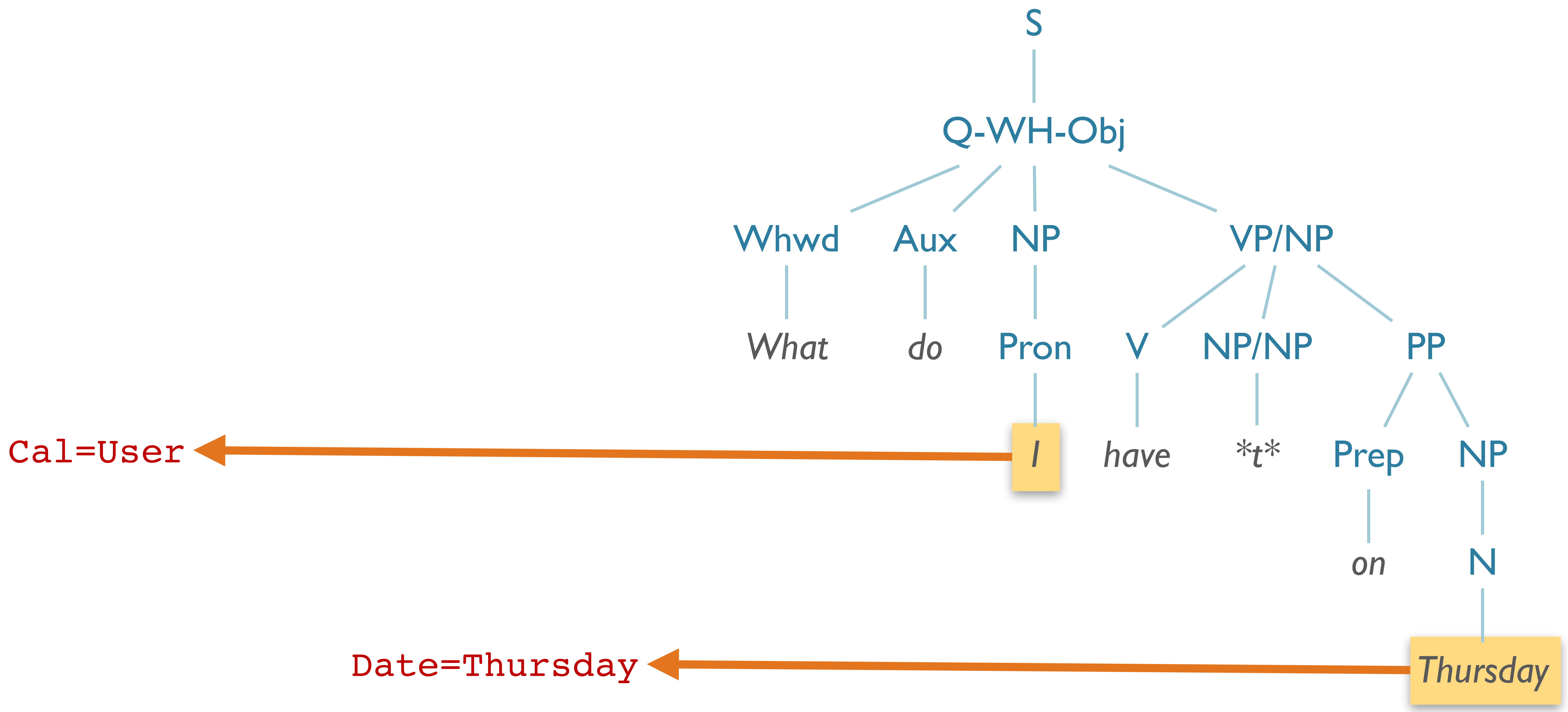
- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!
  - Here's the structure!
- System:
  - Great, but what do I *DO* now?
- Need to associate meaning w/structure



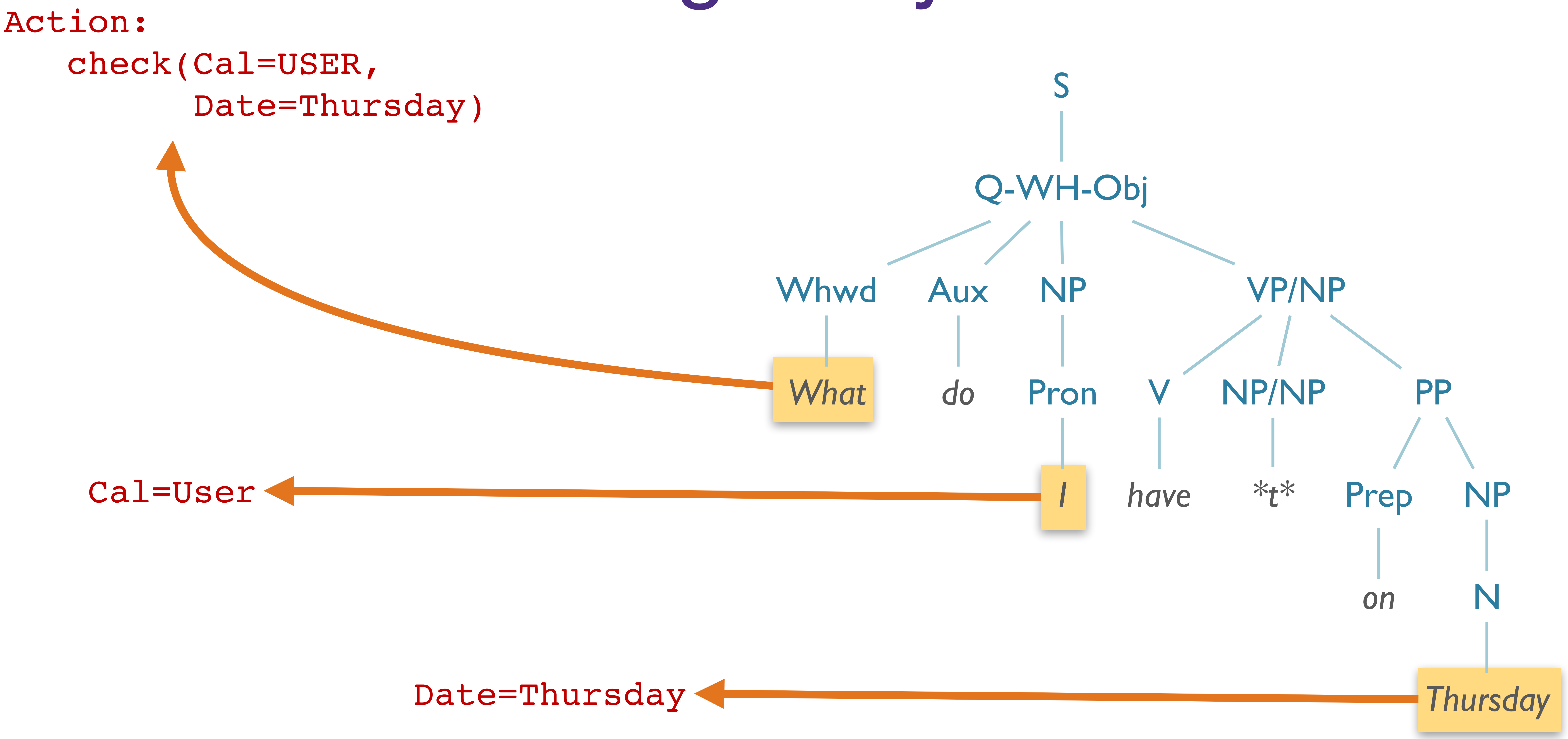
Dialogue System



Dialogue System



Dialogue System



Syntax vs. Semantics

- Syntax:
  - Determine the ***structure*** of natural language input

Syntax vs. Semantics

- Syntax:
  - Determine the ***structure*** of natural language input
- Semantics:
  - Determine the ***meaning*** of natural language input

High-Level Overview

- Semantics = meaning

High-Level Overview

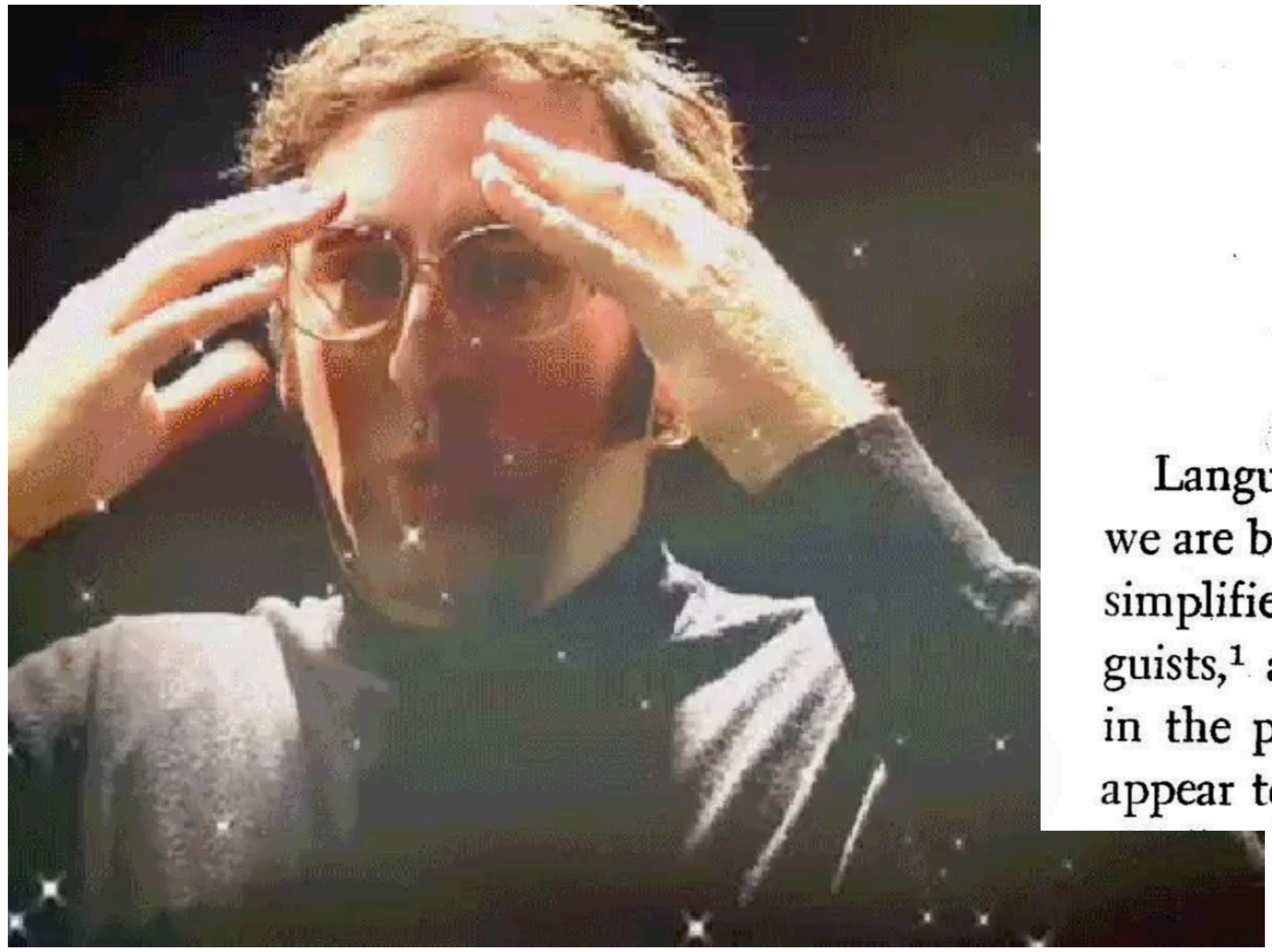
- Semantics = meaning
  - …but what does “meaning” mean?

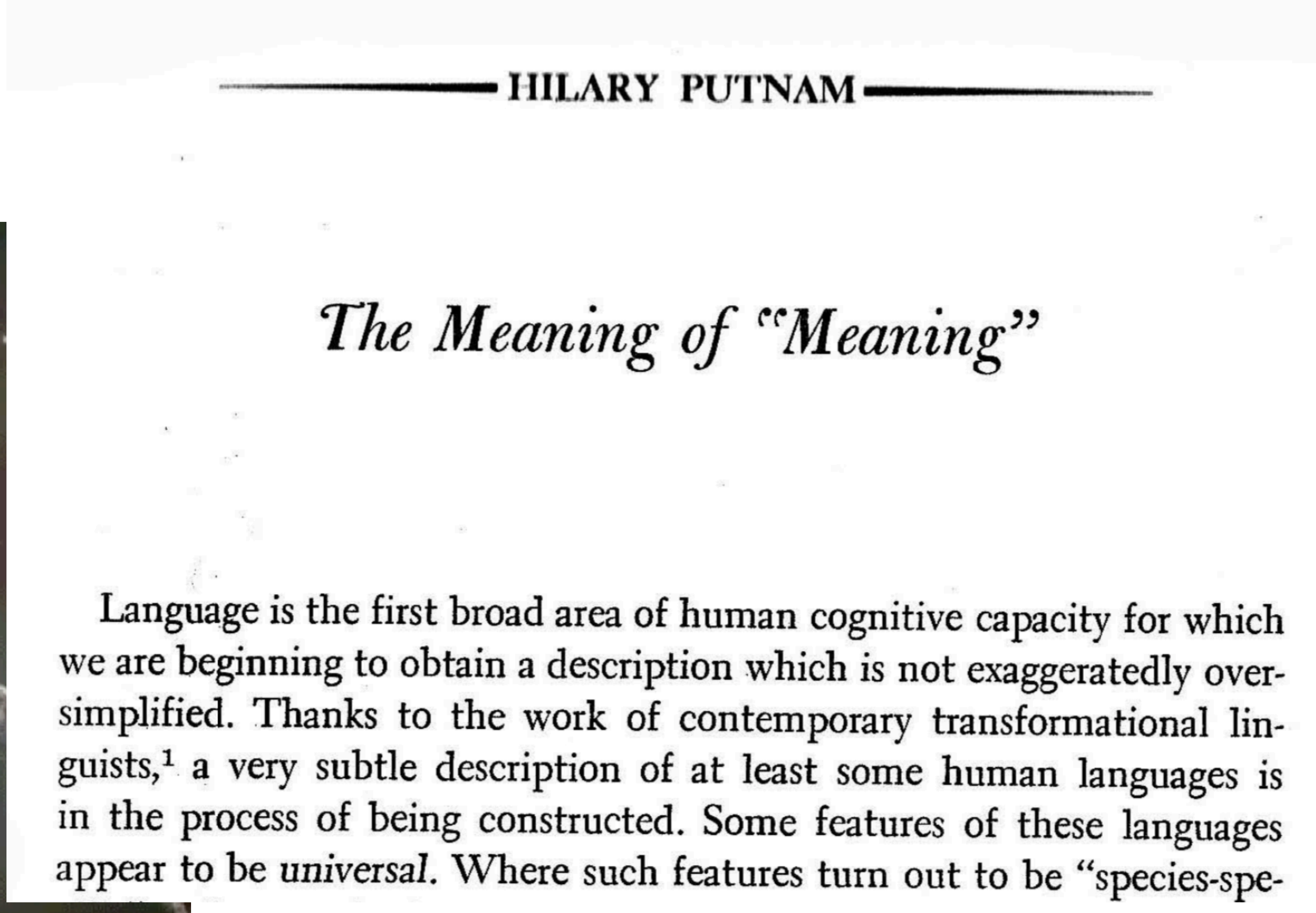
High-Level Overview

- Semantics = meaning
  - …but what does “meaning” mean?

High-Level Overview

- Semantics = meaning
  - …but what does “meaning” mean?





HILARY PUTNAM

The Meaning of “Meaning”

Language is the first broad area of human cognitive capacity for which we are beginning to obtain a description which is not exaggeratedly oversimplified. Thanks to the work of contemporary transformational linguists,[1] a very subtle description of at least some human languages is in the process of being constructed. Some features of these languages appear to be *universal*. Where such features turn out to be “species-spe-

“The sky is blue.”

Speech & Text

"The sky is blue."

Speech & Text

$$\exists x\ Sky(x) \wedge Blue(x)$$

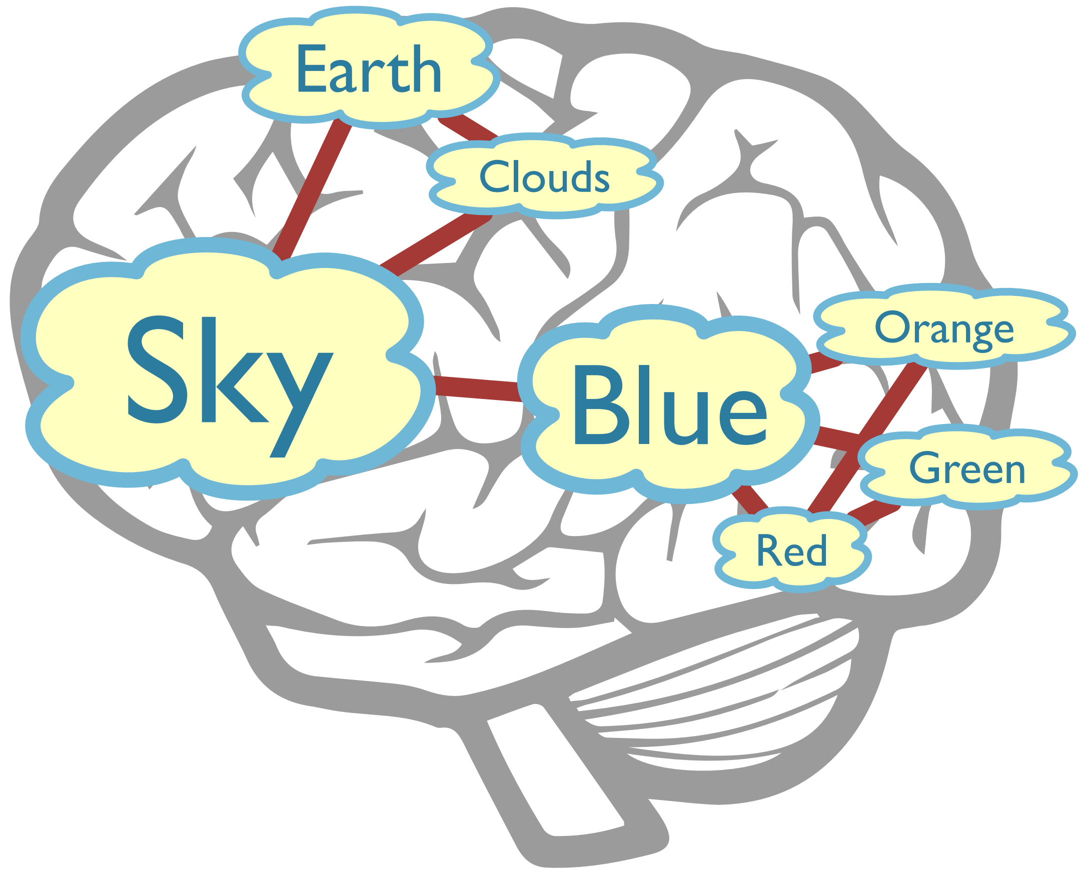
Logic

“The sky is blue.”

Speech & Text

$$\exists x \; Sky(x) \wedge Blue(x)$$

Logic



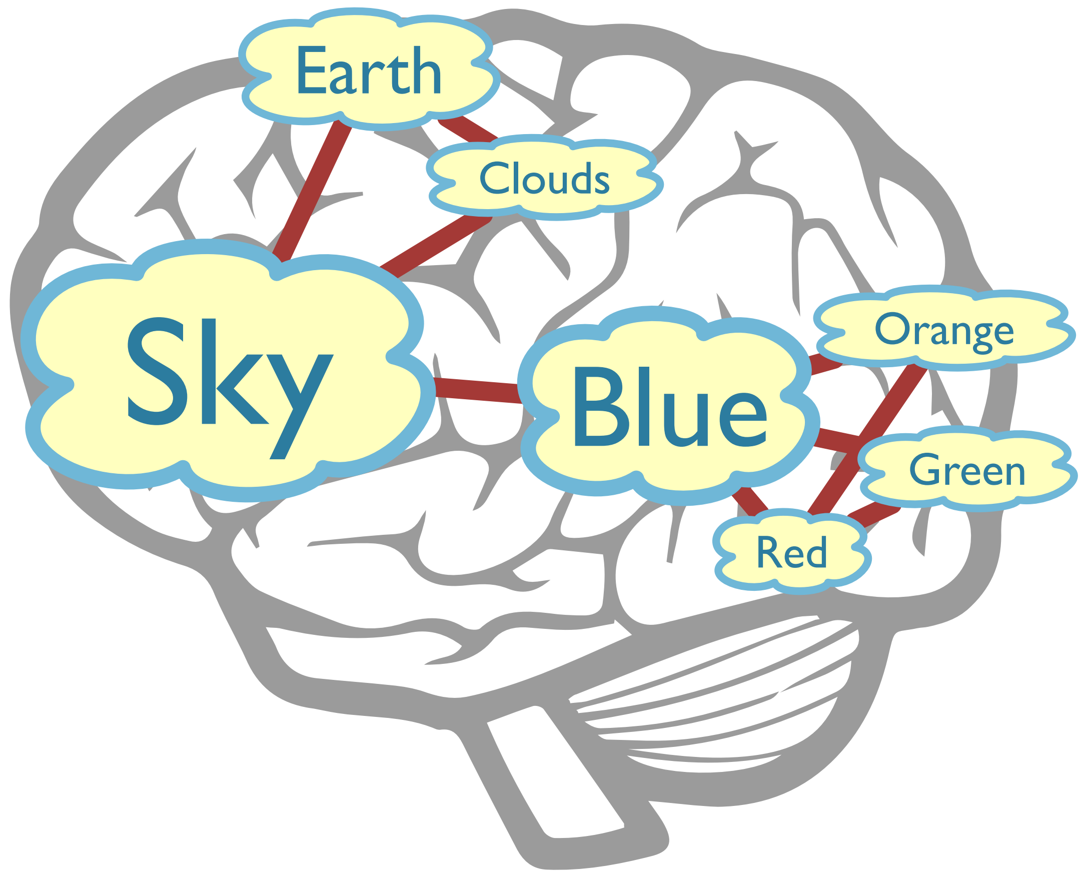
Psychology

"The sky is blue."

Speech & Text

$$\exists x \; Sky(x) \wedge Blue(x)$$

Logic



Psychology

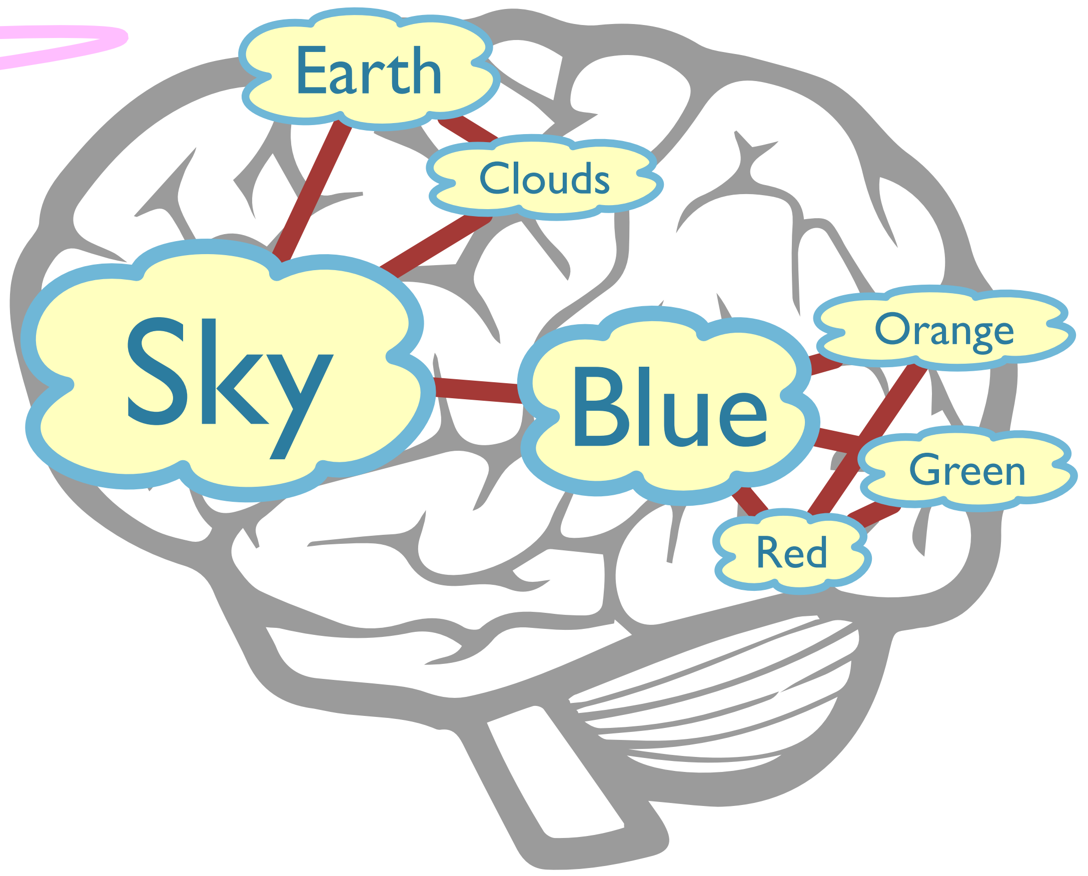
Epistemology

“The sky is blue.”

Speech & Text

$\exists x\ Sky(x) \land Blue(x)$

Logic



Psychology

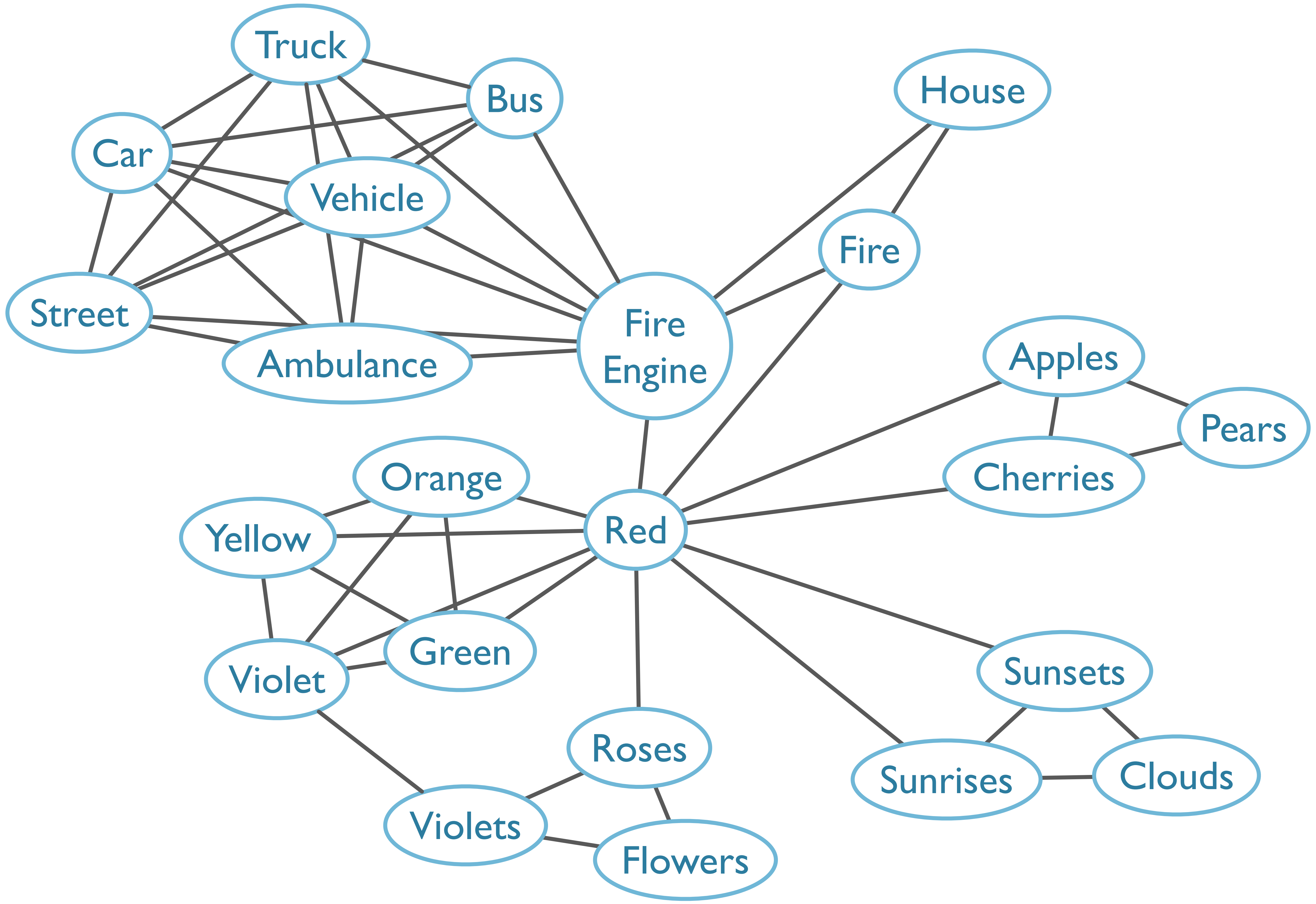
Epistemology

We Will Focus On:

- Concepts that we believe to be true about the world.
- How to connect strings and those concepts.

We ***Won't*** Focus On:

1. Building knowledge bases / semantic networks



Roadmap

- Computational Semantics
  - Overview
  - **Semantics**
  - Representing Meaning
    - First-Order Logic
    - Events
- HW#5
  - Feature grammars in NLTK
  - Practice with animacy

Semantics: an Introduction

Uses for Semantics

- Semantic interpretation required for many tasks
  - Answering questions
  - Following instructions in a software manual
  - Following a recipe
- Requires more than phonology, morphology, syntax
- Must link linguistic elements to world knowledge

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.
  - There was a confrontation between two groups.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.
  - There was a confrontation between two groups.
  - Anti-government crowds are not Mubarak supporters

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.
  - There was a confrontation between two groups.
  - Anti-government crowds are not Mubarak supporters
  - …etc.

Challenges in Semantics

- **Semantic Representation**:
  - What is the appropriate formal language to express propositions in linguistic input?
  - e.g.: predicate calculus: $\exists x \left(dog\left(x\right) \wedge disappear\left(x\right) \right)$

Challenges in Semantics

- **Semantic Representation**:
  - What is the appropriate formal language to express propositions in linguistic input?
  - e.g.: predicate calculus: $\exists x \left(dog(x) \wedge disappear(x) \right)$
- **Entailment**:
  - What are all the conclusions that can be validly drawn from a sentence?
    - *Lincoln was assassinated* $\vDash$ *Lincoln is dead*
    - $\vDash$ "semantically entails": if former is true, the latter must be too

Challenges in Semantics

- **Reference**
  - How do linguistic expressions link to objects/concepts in the real world?
    - ‘the dog,’ ‘the evening star,’ ‘The Superbowl’

Challenges in Semantics

- **Reference**
  - How do linguistic expressions link to objects/concepts in the real world?
    - ‘the dog,’ ‘the evening star,’ ‘The Superbowl’
- **Compositionality**
  - How can we derive the meaning of a unit from its parts?
  - How do syntactic structure and semantic composition relate?
  - ‘rubber duck’ vs. ‘rubber chicken’ vs. ‘rubber-neck’
  - *kick the bucket*

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.

- Define a **meaning representation**

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.
- Define a **meaning representation**
- Develop techniques for **semantic analysis**
  - …convert strings from natural language to meaning representations

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.

- Define a **meaning representation**
- Develop techniques for **semantic analysis**
  - …convert strings from natural language to meaning representations
- Develop methods for **reasoning** about these representations
  - …and performing inference

Tasks in Computational Semantics

- Semantic similarity (words, texts)
- Semantic role labeling
- Semantic analysis / semantic “parsing”
- Recognizing textual entailment (RTE) / natural language inference (NLI)
- Sentiment analysis

Complexity of Computational Semantics

- Knowledge of **language**
  - words, syntax, relationships between structure & meaning, composition procedures

Complexity of Computational Semantics

- Knowledge of **language**
  - words, syntax, relationships between structure & meaning, composition procedures
- Knowledge of **the world**:
  - what are the objects that we refer to?
  - How do they relate?
  - What are their properties?

Complexity of Computational Semantics

- Knowledge of **language**
  - words, syntax, relationships between structure & meaning, composition procedures
- Knowledge of **the world**:
  - what are the objects that we refer to?
  - How do they relate?
  - What are their properties?
- **Reasoning**
  - Given a representation and world, what new conclusions (bits of meaning) can we infer?

Complexity of Computational Semantics

- Effectively AI-complete
  - Needs representation, reasoning, world model, etc.

Representing Meaning

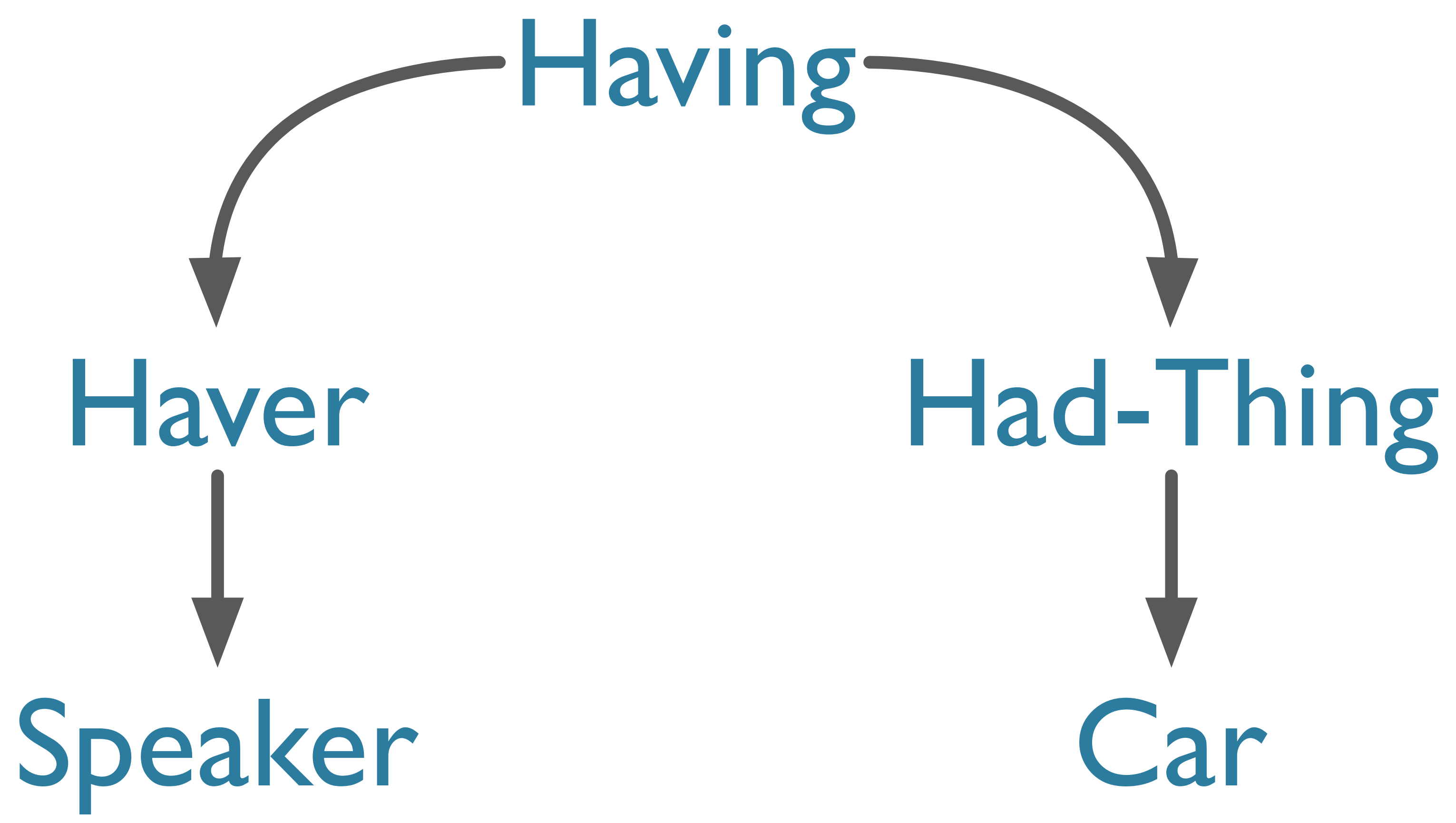
"I have a car"

First-Order Logic: $\exists e, y \left(Having\,(e) \wedge Haver\,(e, Speaker) \wedge HadThing\,(e, y) \wedge Car\,(y) \right)$

“I have a car”

First-Order Logic: $\exists e, y \left(Having\,(e) \wedge Haver\,(e, Speaker) \wedge HadThing\,(e, y) \wedge Car\,(y) \right)$

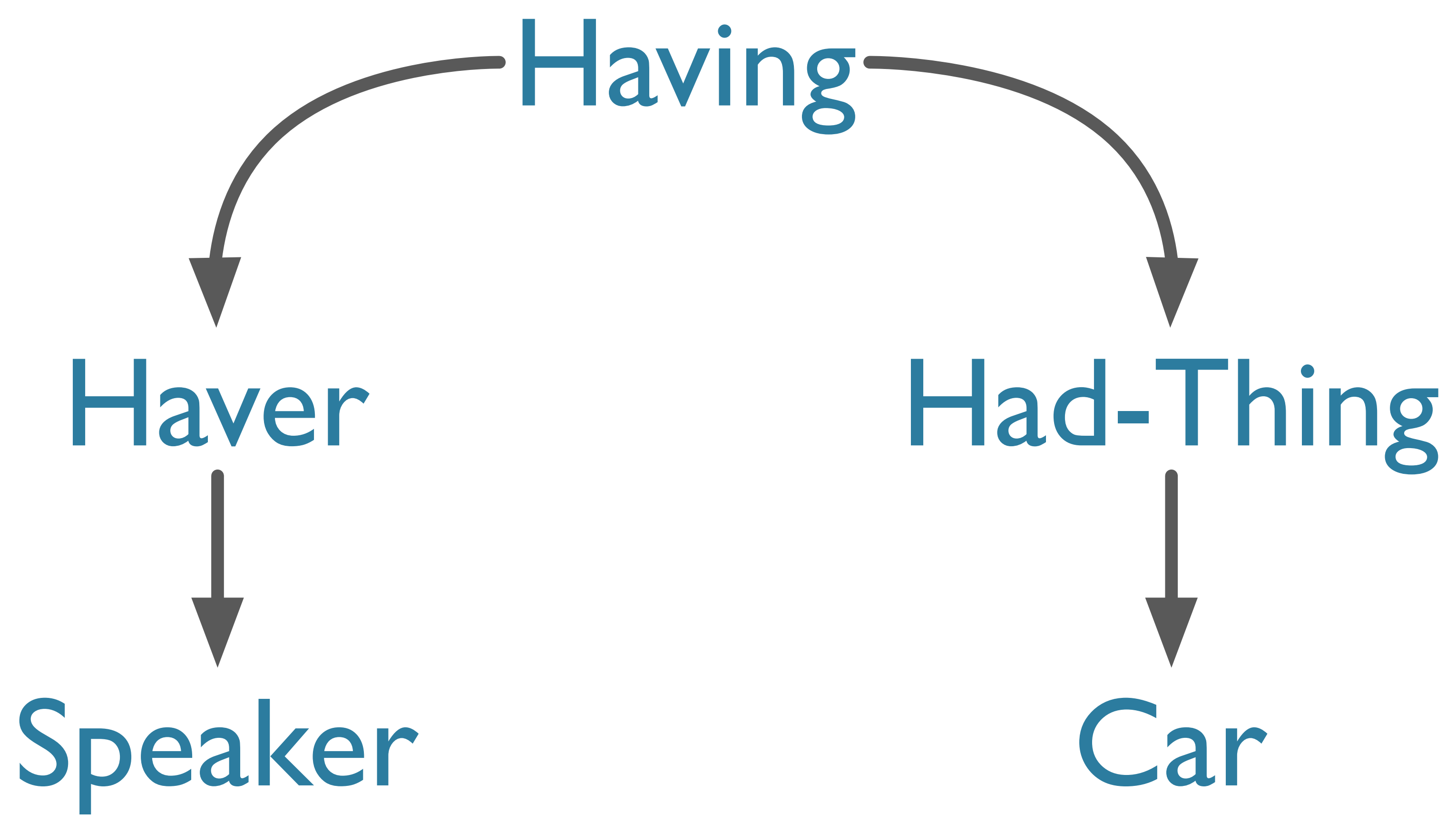
Semantic Network:



"I have a car"

First-Order Logic: $\exists e, y \left(Having\,(e) \wedge Haver\,(e, Speaker) \wedge HadThing\,(e, y) \wedge Car\,(y) \right)$

Semantic Network:



Conceptual Dependency:

$$\begin{array}{l} \textit{Car} \\ \Uparrow \ \textsc{Poss-By} \\ \textit{Speaker} \end{array}$$

"I have a car"

First-Order Logic: $\exists e, y \left(Having\,(e) \land Haver\,(e, Speaker) \land HadThing\,(e, y) \land Car\,(y) \right)$

Semantic Network:

$$\begin{tikzcd}
& \text{Having} \arrow[dl] \arrow[dr] & \\
\text{Haver} \arrow[d] & & \text{Had-Thing} \arrow[d] \\
\text{Speaker} & & \text{Car}
\end{tikzcd}$$

Conceptual Dependency:

$$\begin{array}{l} \quad \textit{Car} \\ \quad \Uparrow \textsc{Poss-By} \\ \textit{Speaker} \end{array}$$

Frame-Based:

```
Having
    Haver: Speaker
    HadThing: Car
```

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary
- Symbol structures correspond to:
  - Objects
  - Properties of objects
  - Relations among objects

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary
- Symbol structures correspond to:
  - Objects
  - Properties of objects
  - Relations among objects
- Can be viewed as:
  - Representation of meaning of linguistic input
  - Representation of state of world

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary
- Symbol structures correspond to:
  - Objects
  - Properties of objects
  - Relations among objects
- Can be viewed as:
  - Representation of meaning of linguistic input
  - Representation of state of world
- Here we focus on **literal** meaning ("what is said")

Representational Requirements

- Verifiability
- Unambiguous representations
- Canonical Form
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
- Canonical Form
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
  - Alternate expressions of same meaning map to same representation
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
  - Alternate expressions of same meaning map to same representation
- Inference and Variables
  - Way to draw valid conclusions from semantics and KB
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
  - Alternate expressions of same meaning map to same representation
- Inference and Variables
  - Way to draw valid conclusions from semantics and KB
- Expressiveness
  - Represent any natural language utterance

Meaning Structure of Language

- Human Languages:
  - Display basic predicate-argument structure
  - Employ variables
  - Employ quantifiers
  - Exhibit a (partially) compositional semantics

Predicate-Argument Structure

- Represent concepts and relationships

Predicate-Argument Structure

- Represent concepts and relationships
- Some words behave like predicates
  - $\boldsymbol{Book}(John,\ United)$; $\boldsymbol{Non\text{-}stop}(Flight)$

Predicate-Argument Structure

- Represent concepts and relationships
- Some words behave like predicates
  - $\boldsymbol{Book}(John,\ United);\ \boldsymbol{Non\text{-}stop}(Flight)$
- Some words behave like arguments
  - $Book(\boldsymbol{John},\ \boldsymbol{United});\ Non\text{-}stop(\boldsymbol{Flight})$

Predicate-Argument Structure

- Represent concepts and relationships
- Some words behave like predicates
  - $\boldsymbol{Book}(John,\ United);\ \boldsymbol{Non\text{-}stop}(Flight)$
- Some words behave like arguments
  - $Book(\boldsymbol{John},\ \boldsymbol{United});\ Non\text{-}stop(\boldsymbol{Flight})$
- Subcategorization frames indicate:
  - Number, Syntactic category, order of args, possibly other features of args

First-Order Logic

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth
- Supports compositionality of meaning*

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth
- Supports compositionality of meaning*
- Supports inference

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth
- Supports compositionality of meaning*
- Supports inference
- Supports generalization through variables

First-Order Logic Terms

- **Constants**: specific objects in world;
  - $A,\ B,\ John$
  - Refer to exactly one object
  - Each object can have multiple constants refer to it
    - $WAStateGovernor$ and $JayInslee$

First-Order Logic Terms

- **Constants**: specific objects in world;
  - $A,\ B,\ John$
  - Refer to exactly one object
  - Each object can have multiple constants refer to it
    - $WAStateGovernor$ and $JayInslee$
- **Functions**: concepts relating ***objects*** → ***objects***
  - $GovernerOf(WA)$
  - Refer to objects, avoid using constants

First-Order Logic Terms

- **Constants**: specific objects in world;
  - $A, \ B, \ John$
  - Refer to exactly one object
  - Each object can have multiple constants refer to it
    - $WAStateGovernor$ and $JayInslee$
- **Functions**: concepts relating ***objects*** → ***objects***
  - $GovernerOf(WA)$
  - Refer to objects, avoid using constants
- **Variables**:
  - $\boldsymbol{x}, \ \boldsymbol{e}$
  - Refer to any potential object in the world

First-Order Logic Language

- **Predicates**
  - Relate ***objects*** to other ***objects***
  - *'United serves Chicago'*
    - $Serves(United, Chicago)$

First-Order Logic Language

- **Predicates**
  - Relate ***objects*** to other ***objects***
  - *'United serves Chicago'*
    - $Serves(United, Chicago)$
- **Logical Connectives**
  - $\{\wedge, \vee, \Rightarrow\}$ = {and, or, implies}
  - Allow for compositionality of meaning* [* many subtleties]
  - *'Frontier serves Seattle and is cheap.'*
    - $Serves(Frontier, Seattle) \wedge Cheap(Frontier)$

Quantifiers

- $\exists$: **existential** quantifier: *"there exists"*

Quantifiers

- $\exists$: **existential** quantifier: *"there exists"*
- Indefinite NP
  - $\geq$**one** such object required for truth

Quantifiers

- ∃: **existential** quantifier: *"there exists"*
- Indefinite NP
  - ≥**one** such object required for truth
- **A non-stop flight** that **serves Pittsburgh**:

$$\exists x\ Flight(x) \wedge Serves(x, Pittsburgh) \wedge Non\text{-}stop(x)$$

Quantifiers

- $\forall$: **universal** quantifier: *"for all"*
  - **All flights include beverages.**

Quantifiers

- $\forall$: **universal** quantifier: *"for all"*
  - **All flights include beverages.**

    $\forall x \; Flight(x) \Rightarrow Includes(x, \; beverages)$

FOL Syntax Summary

$$
\begin{array}{lcc}
\textbf{\textit{Formula}} & \rightarrow & \textit{AtomicFormula} \\
 & | & \textit{Formula Connective Formula} \\
 & | & \textit{Quantifier Variable, ... Formula} \\
 & | & \neg\ \textit{Formula} \\
 & | & (\textit{Formula}) \\
\textbf{\textit{AtomicFormula}} & \rightarrow & \textit{Predicate}(\textit{Term},\ldots) \\
\textbf{\textit{Term}} & \rightarrow & \textit{Function}(\textit{Term},\ldots) \\
 & | & \textit{Constant} \\
 & | & \textit{Variable}
\end{array}
$$

$$
\begin{array}{lcc}
\textbf{\textit{Connective}} & \rightarrow & \wedge \mid \vee \mid \Rightarrow \\
\textbf{\textit{Quantifier}} & \rightarrow & \forall \mid \exists \\
\textbf{\textit{Constant}} & \rightarrow & \textit{VegetarianFood} \mid \textit{Maharani} \mid \ldots \\
\textbf{\textit{Variable}} & \rightarrow & x \mid y \mid \ldots \\
\textbf{\textit{Predicate}} & \rightarrow & \textit{Serves} \mid \textit{Near} \mid \ldots \\
\textbf{\textit{Function}} & \rightarrow & \textit{LocationOf} \mid \textit{CuisineOf} \mid \ldots
\end{array}
$$

J&M p. 556 (3rd ed. 16.3)

Compositionality

- The meaning of a complex expression is a function of the meaning of its parts, and the rules for their combination.

Compositionality

- The meaning of a complex expression is a function of the meaning of its parts, and the rules for their combination.
- Formal languages **are** compositional.

Compositionality

- The meaning of a complex expression is a function of the meaning of its parts, and the rules for their combination.
- Formal languages **are** compositional.
- Natural language meaning is ***largely compositional,*** though not fully.

Compositionality

- …how can we derive:
  - $loves(John,\ Mary)$

Compositionality

- …how can we derive:
  - $loves(John,\ Mary)$
- from:
  - $John$
  - $loves(x,\ y)$
  - $Mary$

Compositionality

- …how can we derive:
  - $loves(John,\ Mary)$
- from:
  - $John$
  - $loves(x,\ y)$
  - $Mary$
- Lambda expressions!

Lambda Expressions

- Lambda (λ) notation (Church, 1940)
  - Just like lambda in Python, Scheme, etc
  - Allows abstraction over FOL formulae
  - Supports compositionality

- Form: (λ) + variable + FOL expression
  - $\lambda x.P(x)$ "Function taking x to $P(x)$"
  - $\lambda x.P(x)(A) = P(A)$ [called beta-reduction]

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$
$$\lambda x.P(x)(A)$$

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$
$$\lambda x.P(x)(A)$$
$$P(A)$$

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$
$$\lambda x.P(x)(A)$$
$$P(A)$$

- Equivalent to function application

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$

$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$
$$\lambda y.Near(Midway,\ y)(Chicago)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$
$$\lambda y.Near(Midway,\ y)(Chicago)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$
$$\lambda y.Near(Midway,\ y)(Chicago)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$
$$\lambda y.Near(Midway,\ y)(Chicago)$$
$$Near(Midway,\ Chicago)$$

Nested λ-Reduction

- If it helps, think of λs as binding sites:

Nested λ-Reduction

- If it helps, think of λs as binding sites:

Chicago

$$\lambda y.Near(x,\ y)$$

Midway

Nested λ-Reduction

- If it helps, think of λs as binding sites:

Chicago

$$\boldsymbol{Near}(\boldsymbol{x},\ \boldsymbol{y})$$

Midway

Lambda Expressions

- ***Currying***
  - Converting multi-argument predicates to sequence of single argument predicates
  - Why?
    - Incrementally accumulates multiple arguments spread over different parts of parse tree

Lambda Expressions

- ***Currying***
  - Converting multi-argument predicates to sequence of single argument predicates
  - Why?
    - Incrementally accumulates multiple arguments spread over different parts of parse tree

- …or *Schönkfinkelization*

Logical Formulae

- FOL terms (objects): denote elements in a domain

Logical Formulae

- FOL terms (objects): denote elements in a domain
- Atomic formulae are:
  - If properties, sets of domain elements
  - If relations, sets of tuples of elements

Logical Formulae

- FOL terms (objects): denote elements in a domain
- Atomic formulae are:
  - If properties, sets of domain elements
  - If relations, sets of tuples of elements
- Formulae based on logical operators:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
F	F	T	F	F	T
F	T	T	F	T	T
T	F	F	F	T	F
T	T	F	T	T	T

Logical Formulae: Finer Points

- $\vee$ is not exclusive:
  - *Your choice is pepperoni or sausage*
  - …use $\veebar$ or $\oplus$

Logical Formulae: Finer Points

- $\vee$ is not exclusive:
  - *Your choice is pepperoni or sausage*
  - …use $\veebar$ or $\oplus$

- $\Rightarrow$ is the logical form
  - Does not mean the same as natural language "if", just that if LHS=T, then RHS=T

Inference

1. α

Inference

1. α
2. $\alpha \Rightarrow \beta$

Inference

1. α
2. $\alpha \Rightarrow \beta$
3. $\therefore \beta$

Inference

1. $VegetarianRestaurant(Leaf)$

Inference

1. $VegetarianRestaurant(Leaf)$

2. $\forall x \; VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

Inference

1. $VegetarianRestaurant(Leaf)$

2. $\forall x \; VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

3. $\therefore Serves(Leaf, \; VegetarianFood)$

Inference

- Standard AI-type logical inference procedures
  - Modus Ponens
  - Forward-chaining, Backward Chaining
  - Abduction
  - Resolution
  - Etc…

Inference

- Standard AI-type logical inference procedures
  - Modus Ponens
  - Forward-chaining, Backward Chaining
  - Abduction
  - Resolution
  - Etc…
- We'll assume we have a theorem prover.

Roadmap

- Computational Semantics
  - Introduction
  - Semantics
  - Representing Meaning
    - First-Order Logic
    - **Events**
- HW#5
  - Feature grammars in NLTK
  - Practice with animacy

Events

Representing Events

- Initially, single predicate with some arguments
  - $Serves(United,\ Houston)$
  - Assume # of args = # of elements in subcategorization frame

Representing Events

- Initially, single predicate with some arguments
  - $Serves(United, Houston)$
  - Assume # of args = # of elements in subcategorization frame
- Example:
  - *The flight arrived*
  - *The flight arrived in Seattle*
  - *The flight arrived in Seattle on Saturday.*
  - *The flight arrived on Saturday.*
  - *The flight arrived in Seattle from SFO.*
  - *The flight arrived in Seattle from SFO on Saturday.*

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

- *The flight arrived in Seattle from SFO on Saturday.*

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

- *The flight arrived in Seattle from SFO on Saturday.*
  - Davidsonian (Davidson 1967):
    - $\exists \boldsymbol{e}\ Arrival(\boldsymbol{e}, Flight, Seattle, SFO) \land Time(\boldsymbol{e}, Saturday)$

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

- *The flight arrived in Seattle from SFO on Saturday.*
  - Davidsonian (Davidson 1967):
    - $\exists \boldsymbol{e}\ Arrival(\boldsymbol{e}, Flight, Seattle, SFO) \wedge Time(\boldsymbol{e}, Saturday)$
  - Neo-Davidsonian (Parsons 1990):
    - $\exists \boldsymbol{e}\ Arrival(\boldsymbol{e}) \wedge Arrived(\boldsymbol{e}, Flight) \wedge Destination(\boldsymbol{e}, Seattle) \wedge Origin(\boldsymbol{e}, SFO) \wedge Time(\boldsymbol{e}, Saturday)$

Neo-Davidsonian Events

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

Neo-Davidsonian Events

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication
- Pros
  - No fixed argument structure
  - Dynamically add predicates as necessary
  - No unused roles
  - Logical connections can be derived

Meaning Representation for Computational Semantics

- Requirements
  - Verifiability
  - Unambiguous representation
  - Canonical Form
  - Inference
  - Variables
  - Expressiveness
- Solution:
  - First-Order Logic
    - Structure
    - Semantics
    - Event Representation

Summary

- FOL can be used as a meaning representation language for natural language
- Principle of compositionality:
  - The meaning of a complex expression is a function of the meaning of its parts
- λ-expressions can be used to compute meaning representations from syntactic trees based on the principle of compositionality
- In next classes, we will look at syntax-driven approach to semantic analysis in more detail

HW #5: Feature-based Parsing

Agreement with Heads and Features

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

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

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

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

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

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

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

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

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

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.
- Use the grammar to parse a small set of sentences we provide.

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'`

NLTK Feature Syntax

- Basics
  - `X[FEAT₁=VALUE₁, FEAT₂=VALUE₂]`

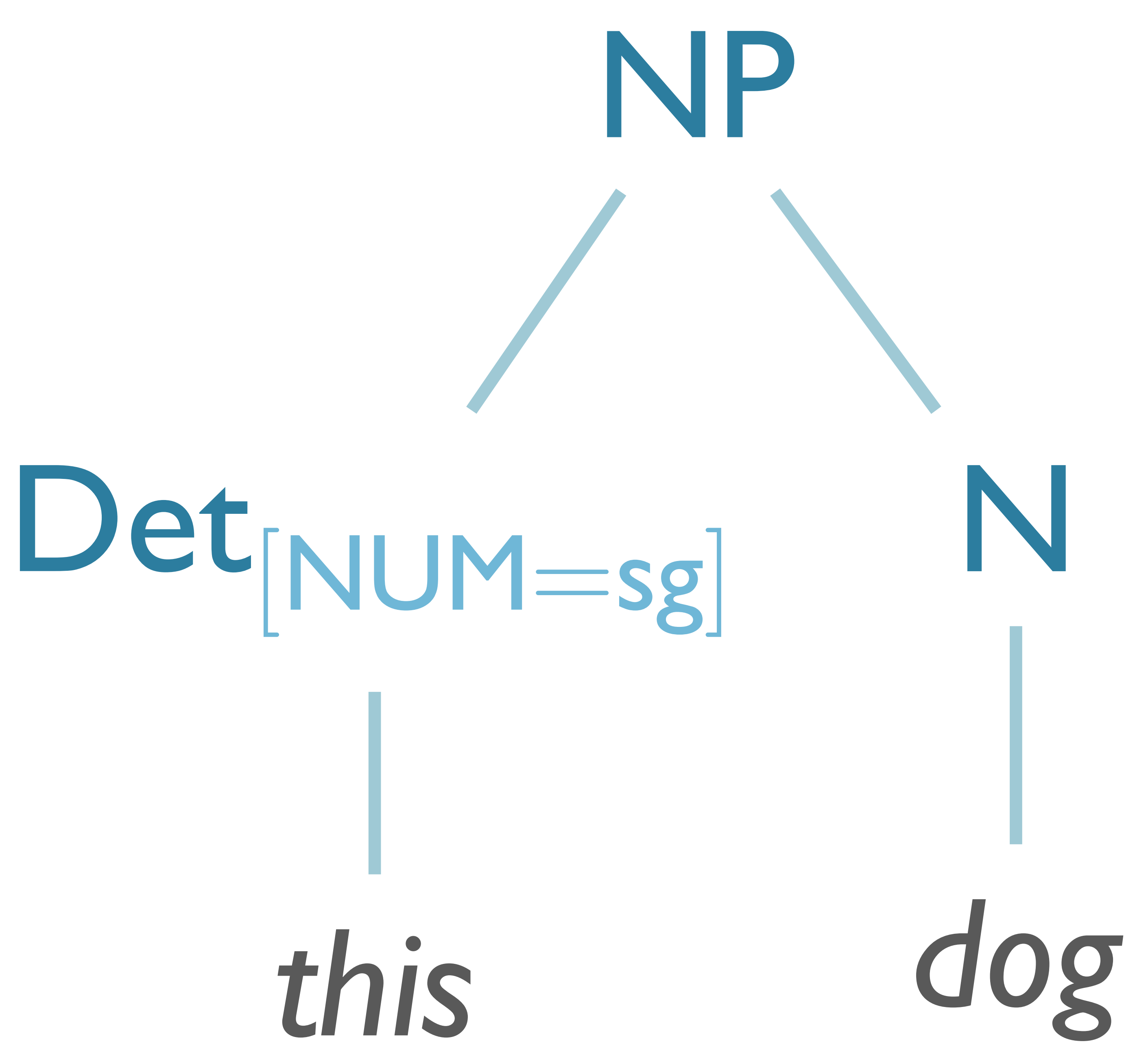
- Variables
  - `X[FEAT=?f]`

- Binary Values
  - `X[-FEAT], Y[+FEAT]`

HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

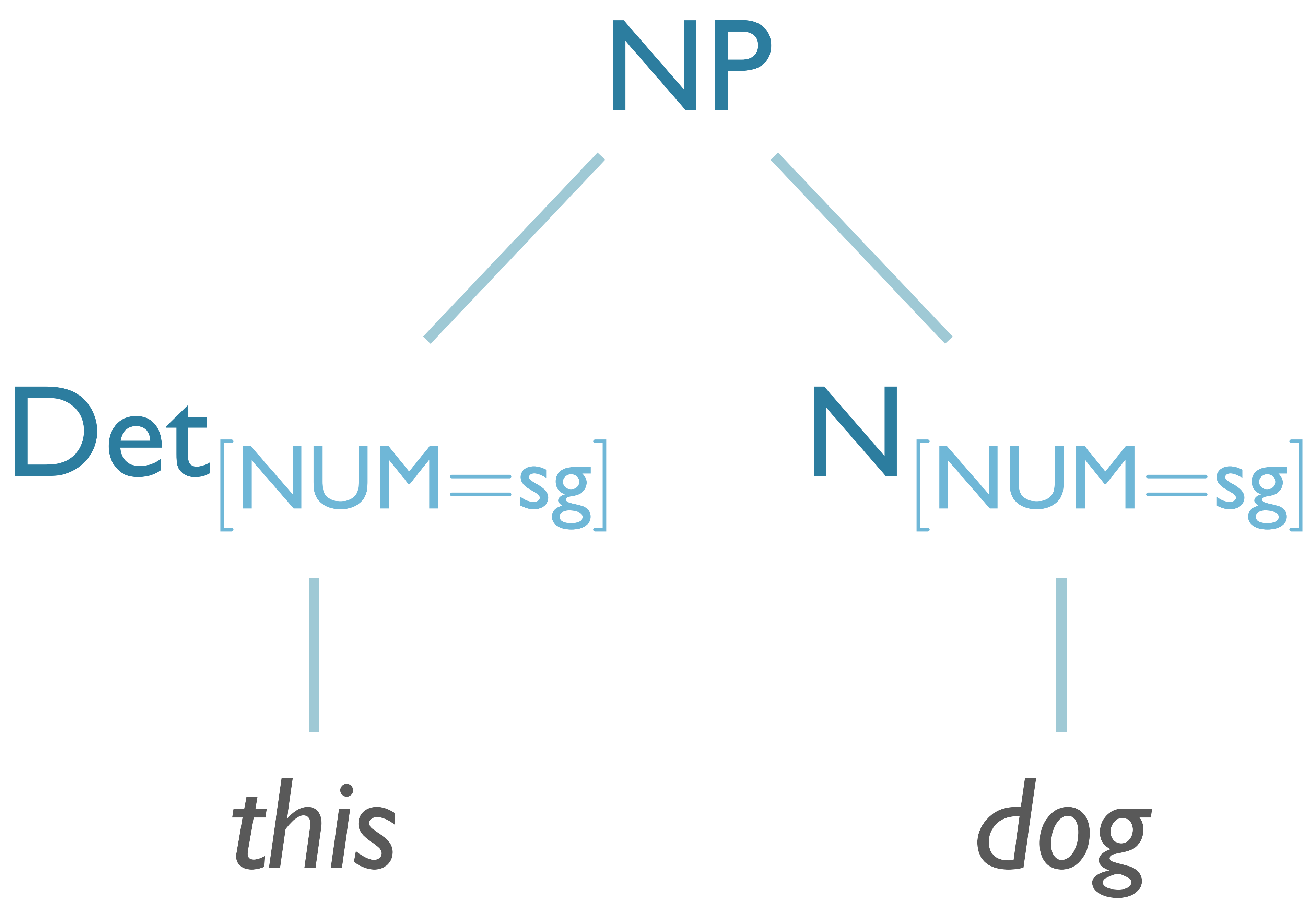
```
Det[NUM=sg] -> 'this' | 'that'
Det[NUM=pl] -> 'these' | 'those'
N[NUM=sg] -> 'dog' | 'cat'
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

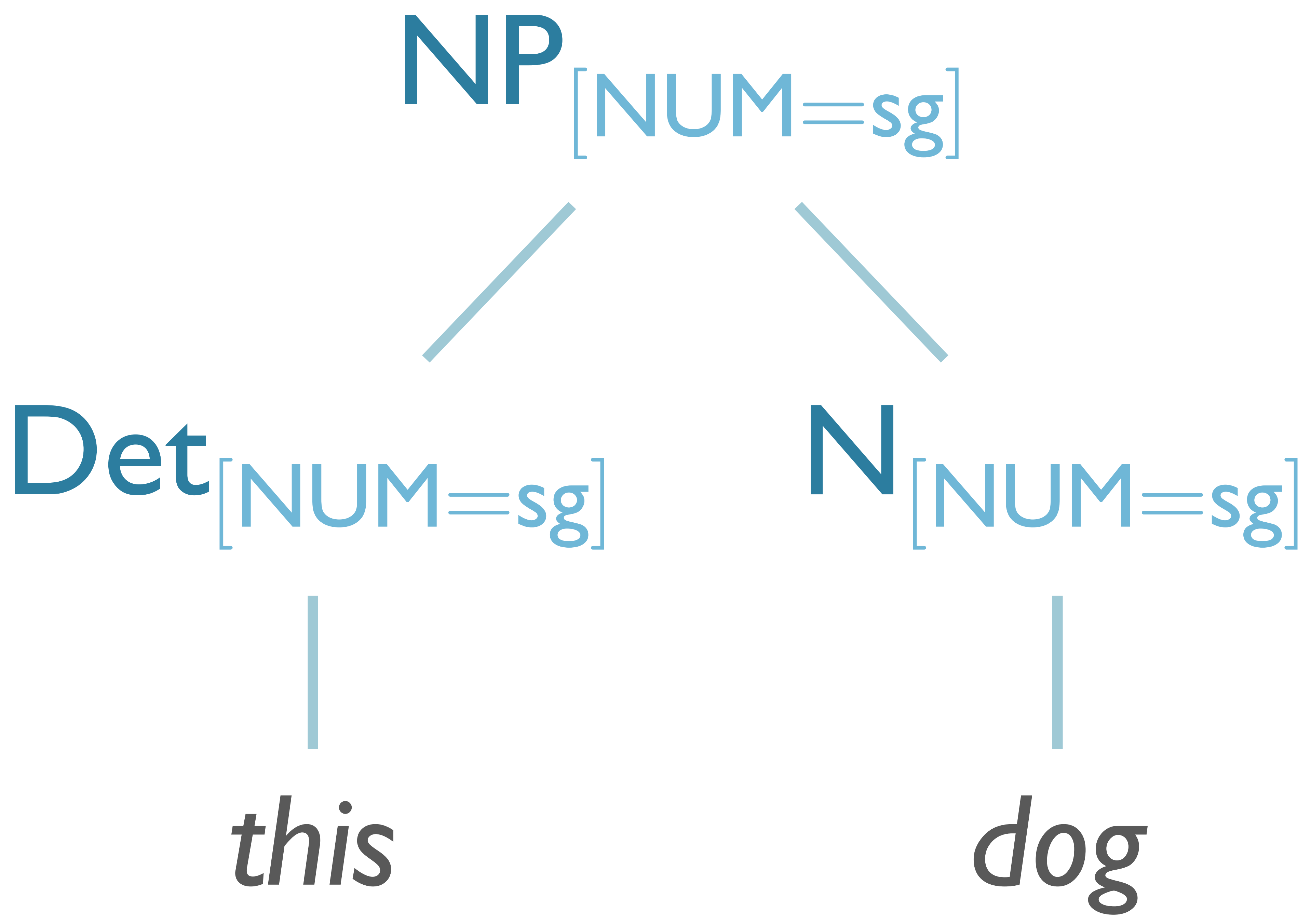
```
Det[NUM=sg] -> 'this' | 'that'
Det[NUM=pl] -> 'these' | 'those'
N[NUM=sg] -> 'dog' | 'cat'
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

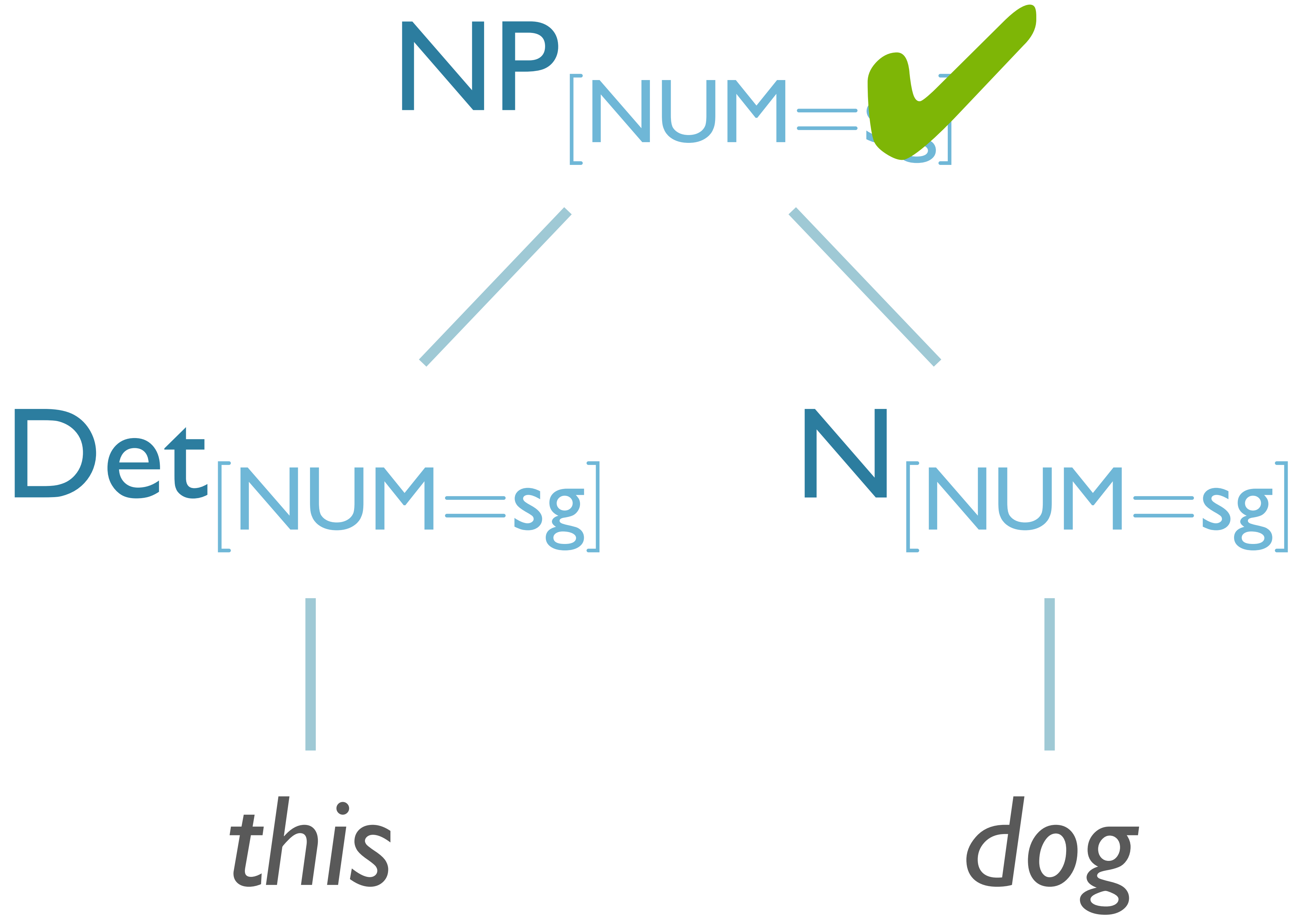
```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

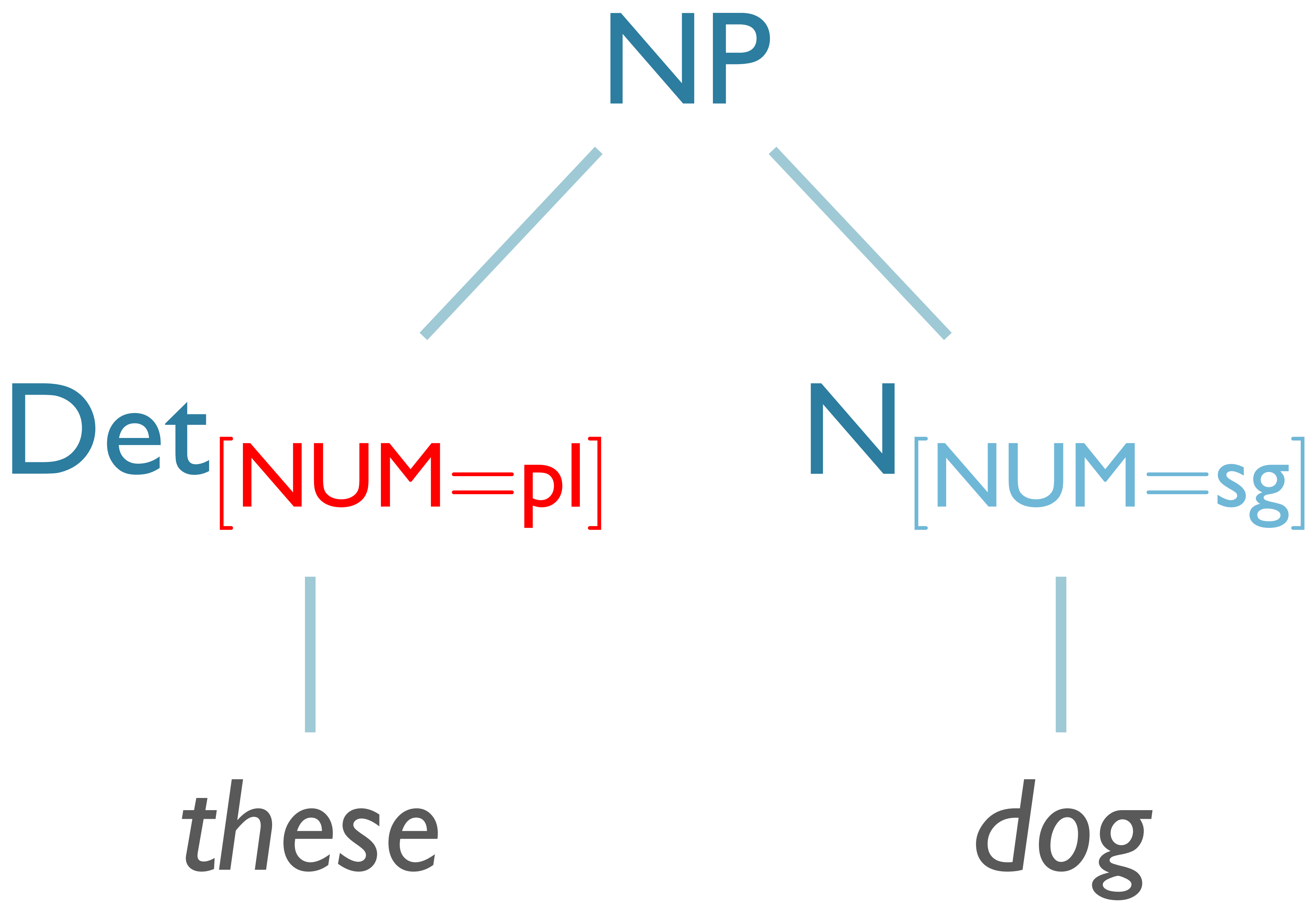
```
Det[NUM=sg] -> 'this' | 'that'
Det[NUM=pl] -> 'these' | 'those'
N[NUM=sg] -> 'dog' | 'cat'
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

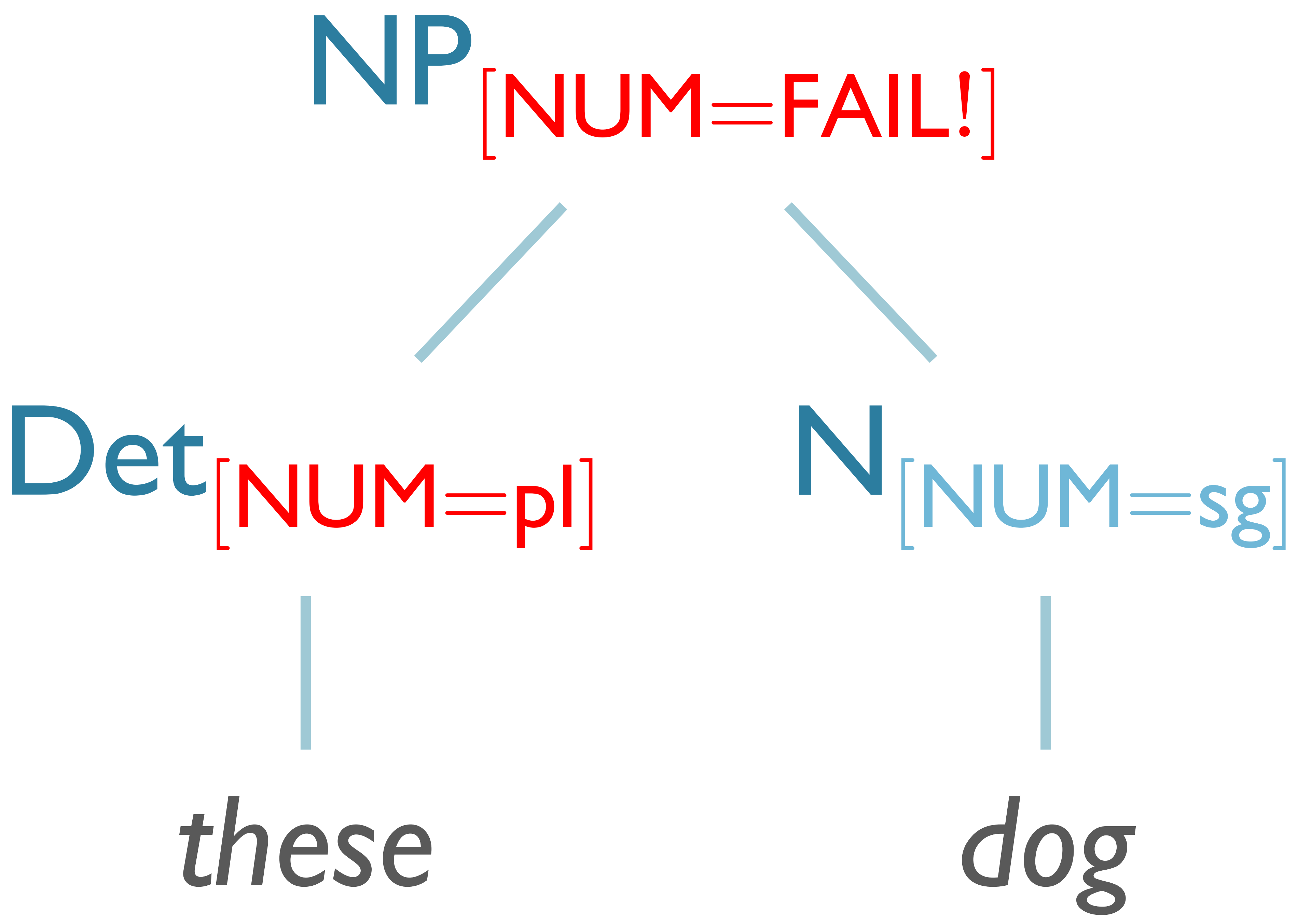
```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



HW #5: NLTK Feature Syntax

```
NP[NUM=?n] -> Det[NUM=?n] N[NUM=?n]
```

```
Det[NUM=sg] -> ‘this’ | ‘that’
Det[NUM=pl] -> ‘these’ | ‘those’
N[NUM=sg] -> ‘dog’ | ‘cat’
```



HW #5: Grammars

- It’s possible to get the grammar to work with completely arbitrary rules, BUT…
- We would prefer them to be linguistically motivated!
  - instead of `[IT_OK=yes]` or `[PRON_AGR=it]`
  - `[GENDER=neut, PERSON=3rd, NUMBER=sg]`

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 (more prominent in other languages):
      - 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].*

Aspect (J&M 17.4.2)

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

Aspect (J&M 17.4.2)

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

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].*

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)

Feature Grammar Practice: Animacy

Feature Grammar Practice

- **Initial Grammar:**

```
S -> NP VP
VP[subcat=ditrans] -> V NP NP
NP -> NNP
NP -> Det N
NNP[animacy=True] -> 'Alex' | 'Ahmed'
V[subcat=ditrans] -> 'gifted'
Det -> 'a' | 'the'
N[animacy=False] -> 'book' | 'rock'
```

Feature Grammar Practice

Feature Grammar Practice

Feature Grammar Practice

Feature Grammar Practice

Practice Task

- Modify the initial grammar to incorporate animacy in such a way that you get the right results:
  - Alex gifted Ahmed a book
  - * Alex gifted the rock a book