# Natural Language Processing

Lecture 19: Compositional Semantics and Semantic Parsing

# Meaning

- The recent lectures dealt with some issues around word meaning, primarily:
  - Relationships between words and word similarity
  - Word sense disambiguation
  - Predicates and their arguments
- Today:  meaning of NL (English) sentences

# Key Challenge of Meaning

- We actually say very little - much more is left unsaid, because it’s assumed to be widely known.

- Examples:
  - Grading assignments
  - Restaurant menus
  - Learning to use a new piece of software

# Meaning Representation Languages

- Symbolic representation that does two jobs:
  - Conveys the meaning of a **sentence**
  - Represents (some part of) the **world**
- Today we'll use **first-order logic**.

# A MRL Should Be Able To …

- Verify a query against a knowledge base
  - Do CMU students follow politics?
- Eliminate ambiguity
  - CMU students enjoy visiting Senators.
- Cope with vagueness
  - Sally heard the news.
- Cope with many ways of expressing the same meaning (canonical forms)
  - The candidate evaded the question.
  - The question was evaded by the candidate.
- Draw conclusions based on the knowledge base
  - Who could become the 48th president?
- Represent all of the meanings we care about

# Model-Theoretic Semantics

- Model:  a simplified representation of the world: objects, properties, relations (**domain**).
- Non-logical vocabulary
  - Each element **denotes** a well-defined part of the model
  - Such a mapping is called an **interpretation**

# A Model

- **Domain**: Noah, Karen, Rebecca, Frederick, Green Mango, Casbah, Udipi, Thai, Mediterranean, Indian
- **Properties**: Green Mango and Udipi are crowded; Casbah is expensive
- **Relations**: Karen likes Green Mango, Frederick likes Casbah, everyone likes Udipi, Green Mango serves Thai, Casbah serves Mediterranean, and Udipi serves Indian
- n, k, r, f, g, c, u, t, m, i
- Crowded = {g, u}
- Expensive = {c}
- Likes = {(k, g), (f, c), (n, u), (k, u), (r, u), (f, u)}
- Serves = {(g, t), (c, m), (u, i)}

# Some English

- Karen likes Green Mango and Frederick likes Casbah.
- Noah and Rebecca like the same restaurants.
- Noah likes expensive restaurants.
- Not everybody likes Green Mango.

- What we want is to be able to represent these statements in a way that lets us compare them to our model.
- **Truth-conditional semantics**: need operators and their meanings, given a particular model.

# First-Order Logic

- **Terms** refer to elements of the domain: **constants**, **functions**, and **variables**
  - Noah, SpouseOf(Karen), X
- **Predicates** are used to refer to sets and relations
  - Serves(Casbah, Mediterranean)
- Logical connectives: ∧ (and), ∨ (or), ¬ (not), ⇒ (implies), ...
- Quantifiers ...

# Quantifiers in FOL

- Two ways to use variables:
  - refer to one anonymous object from the domain (**existential**; ∃; “there exists”)
  - refer to all objects in the domain (**universal**; ∀; “for all”)
- a restaurant near CMU that serves Indian food

  ∃x Restaurant(x), Near(x, CMU), Serves(x, Indian)
- All expensive restaurants are far from campus

  ∀x Restaurant(x), Expensive(x) ⇒ ¬Near(x, CMU)

# Extension: Lambda Notation

- A way of making anonymous functions.
- λx. (*some expression mentioning x*)
  - Example: λx.Near(x, CMU)
  - Deeper example: λx.λy.Serves(y, x)
- Lambda reduction: substitute for the variable.
  - (λx.Near(x, CMU))(Lulu's Noodles)
    becomes
    Near(Lulu's Noodles, CMU)

# Inference

- Big idea: extend the knowledge base, or check some proposition against the knowledge base.
- **Forward chaining** with modus ponens:
  - given $\alpha$ and $\alpha \Rightarrow \beta$, we know $\beta$.
- **Backward chaining** takes a query $\beta$ and looks for propositions $\alpha$ and $\alpha \Rightarrow \beta$ that would prove $\beta$.
  - Not the same as backward reasoning (abduction).
  - Used by Prolog
- Both are sound, neither is complete.

# Lots More To Say About MRLs!

- See chapter 17 for more about:
  - Representing events and states in FOL
  - Dealing with optional arguments (e.g., “eat”)
  - Representing time
  - Non-FOL approaches to meaning

# First-Order Worlds, Then and Now

- Interest in this topic waned during the 1990s and 2000s.
- It’s come back, with the rise of semi-structured databases like Wikipedia.
  - Lay contributors to these databases may be helping us to solve the knowledge acquisition problem.
- Also, lots of research on using NLP, information extraction, and machine learning to grow and improve knowledge bases from free text data.
  - “Read the Web” project here at CMU.

# Connecting Syntax and Semantics

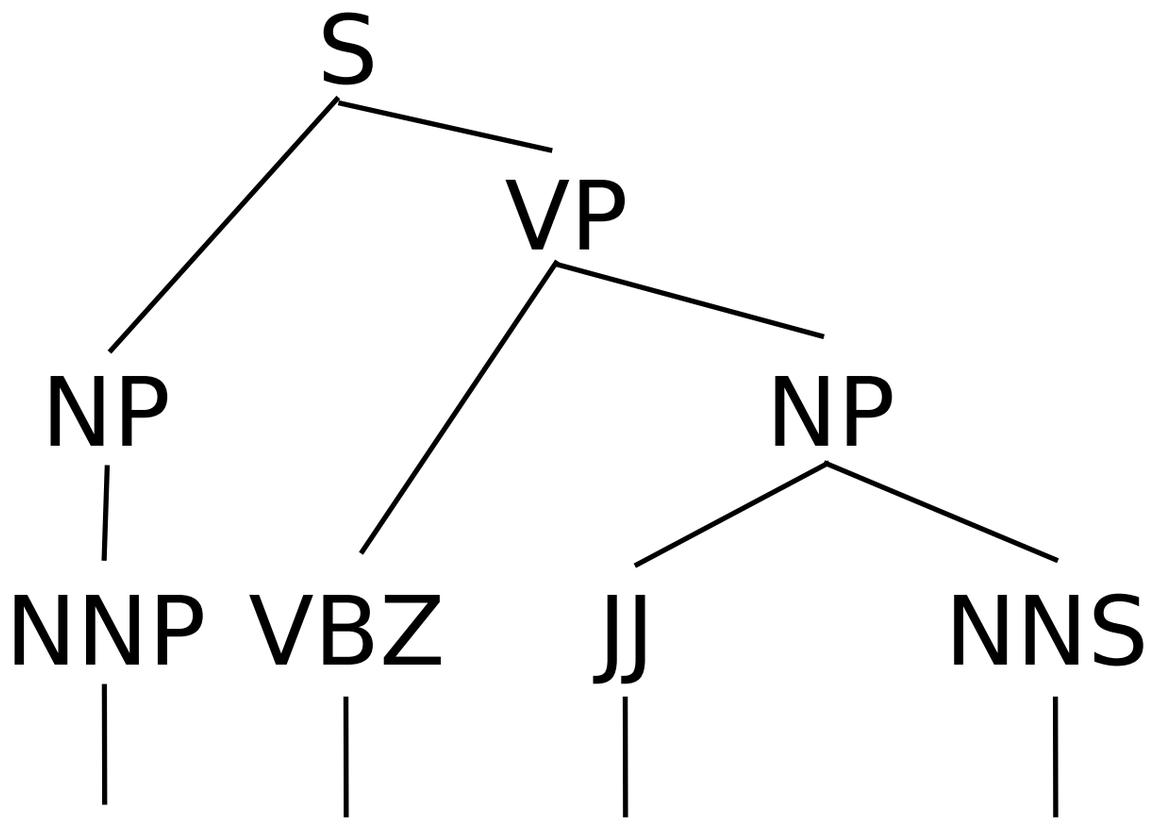
# Semantic Analysis

- Goal:  transform a NL statement into MRL (today, FOL).
- We’re assuming a very literal, inference-free version of meaning!
- Sometimes called “semantic parsing.”

# Compositionality

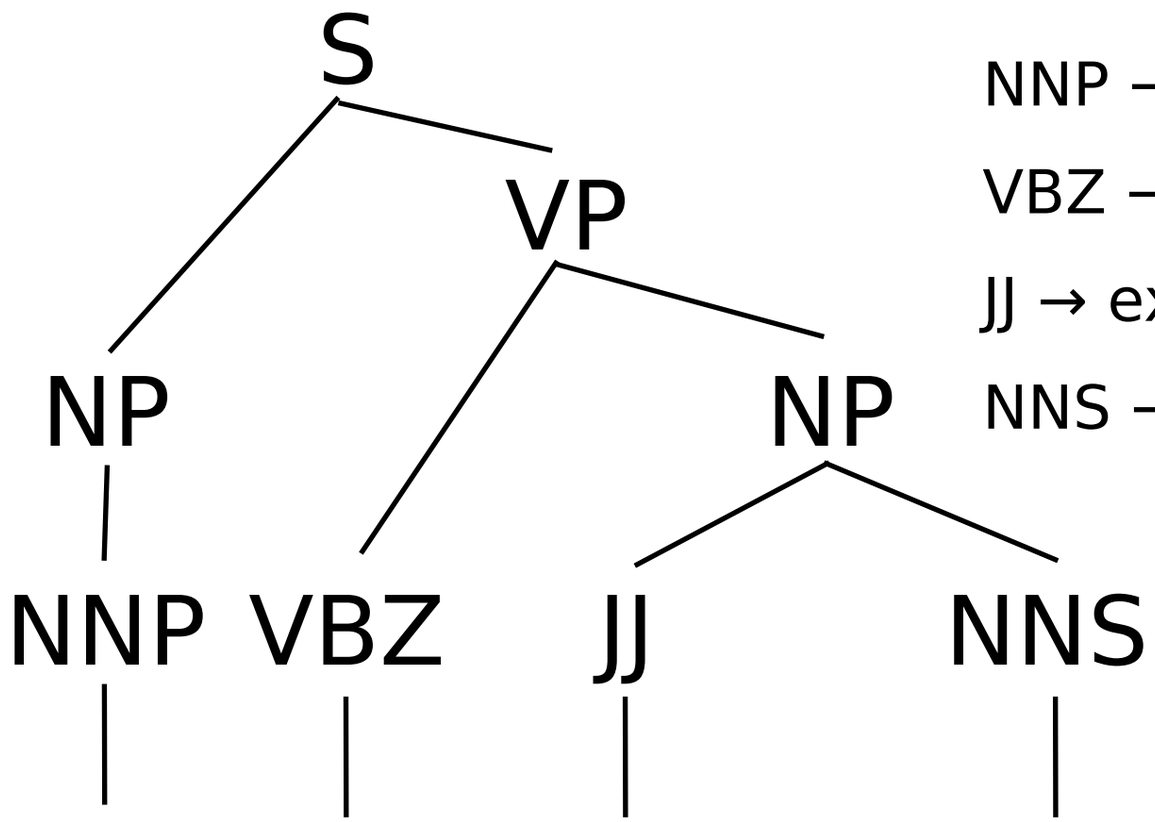
- The meaning of an NL phrase is determined by combining the meaning of its sub-parts.
- There are obvious exceptions (“hot dog,” “straw man,” “New York,” etc.).
- Big idea:  start with parse tree, build semantics on top using FOL with λ-expressions.

# An Example




- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

# An Example




NNP → Noah { Noah }

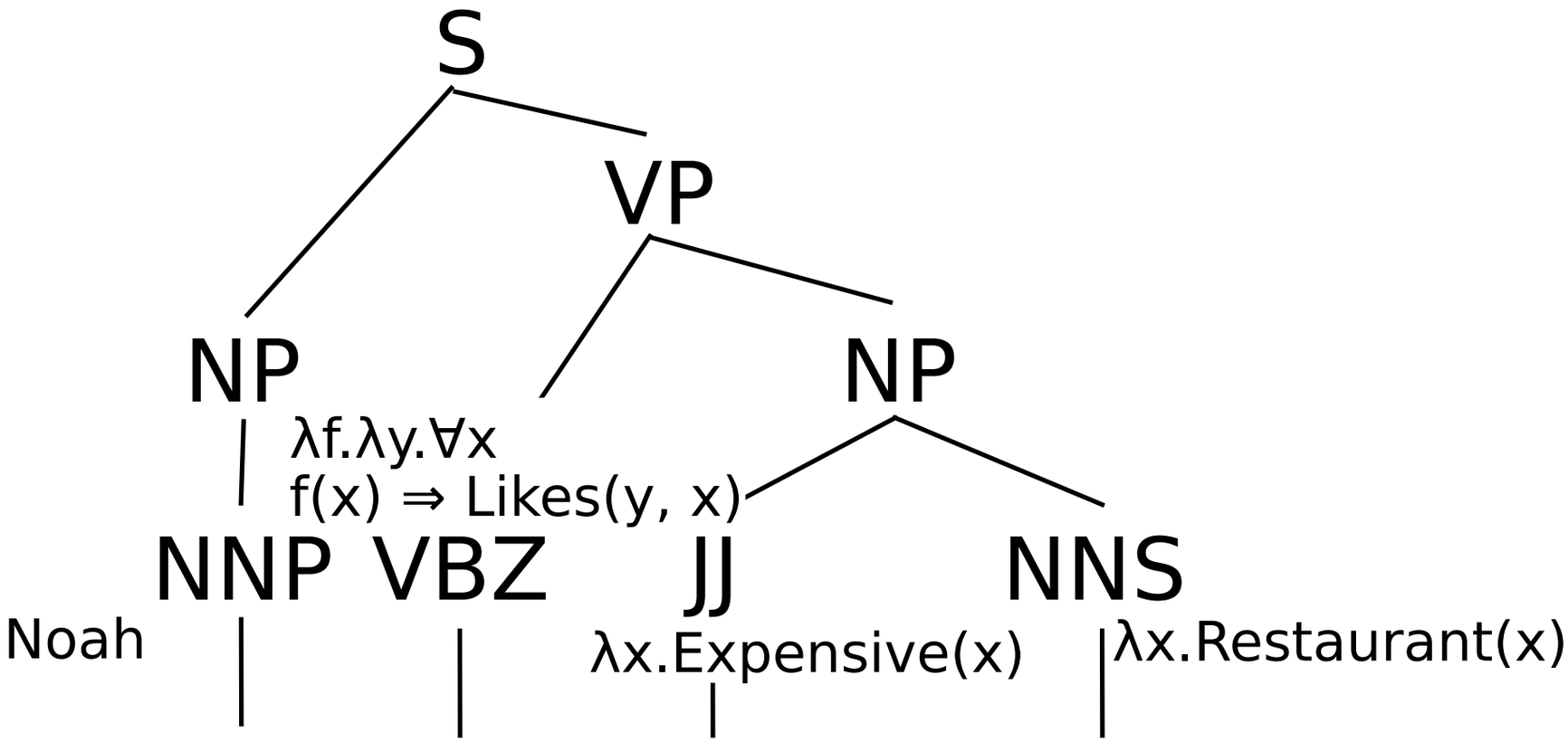
VBZ → likes { λf.λy.∀x f(x) ⇒ Likes(y, x) }

JJ → expensive { λx.Expensive(x) }

NNS → restaurants { λx.Restaurant(x) }

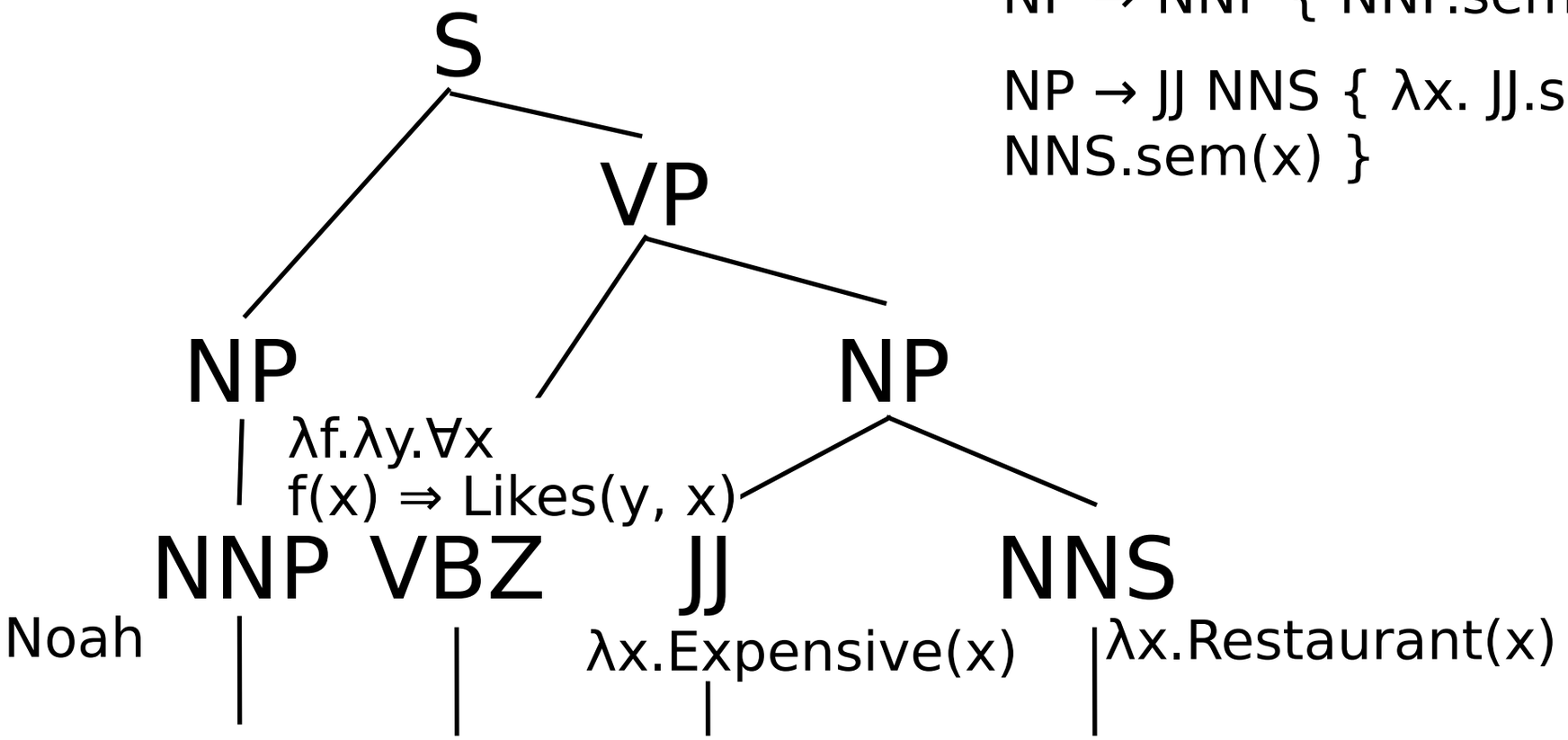
- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

# An Example




- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

# An Example

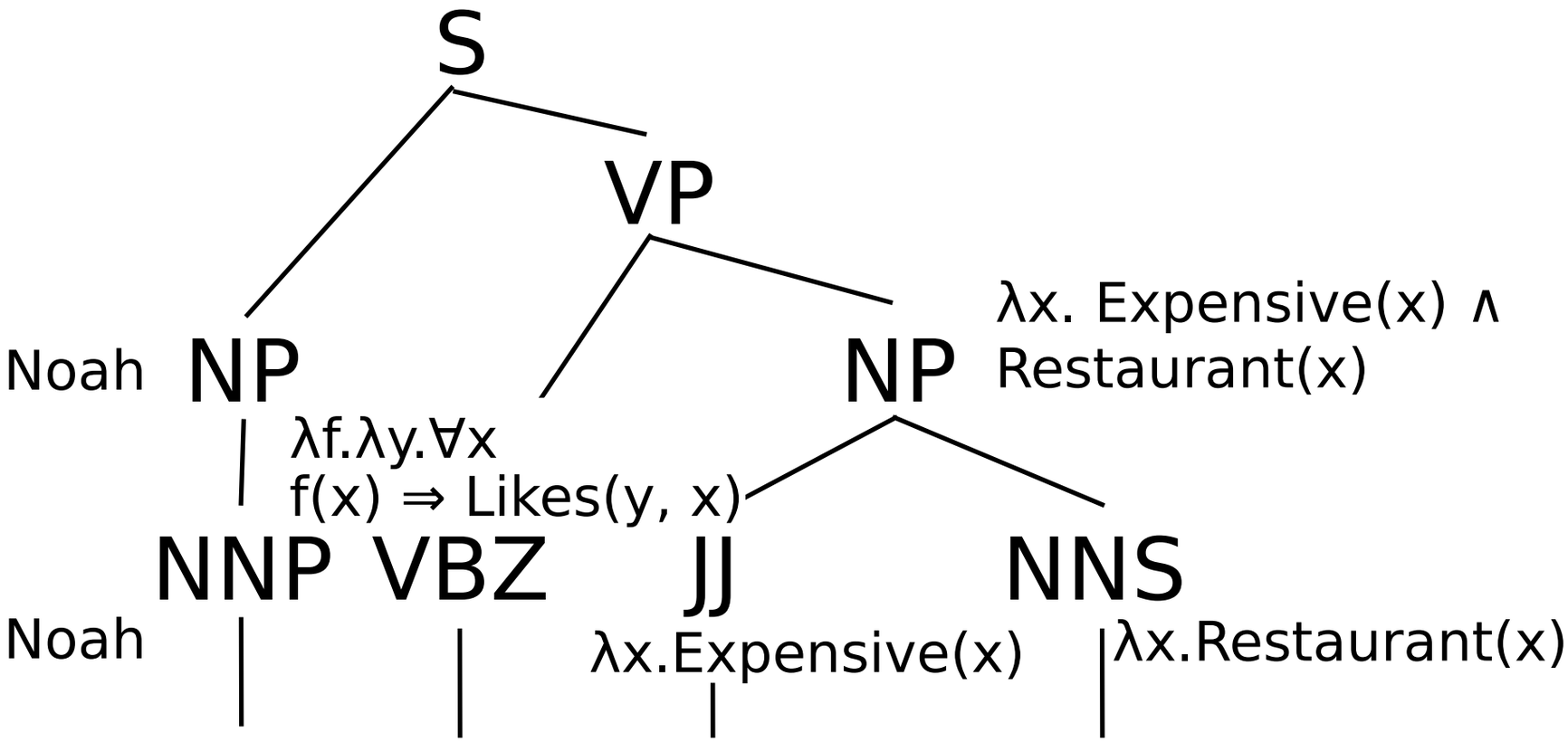



NP → NNP { NNP.sem }

NP → JJ NNS { λx. JJ.sem(x) ∧ NNS.sem(x) }

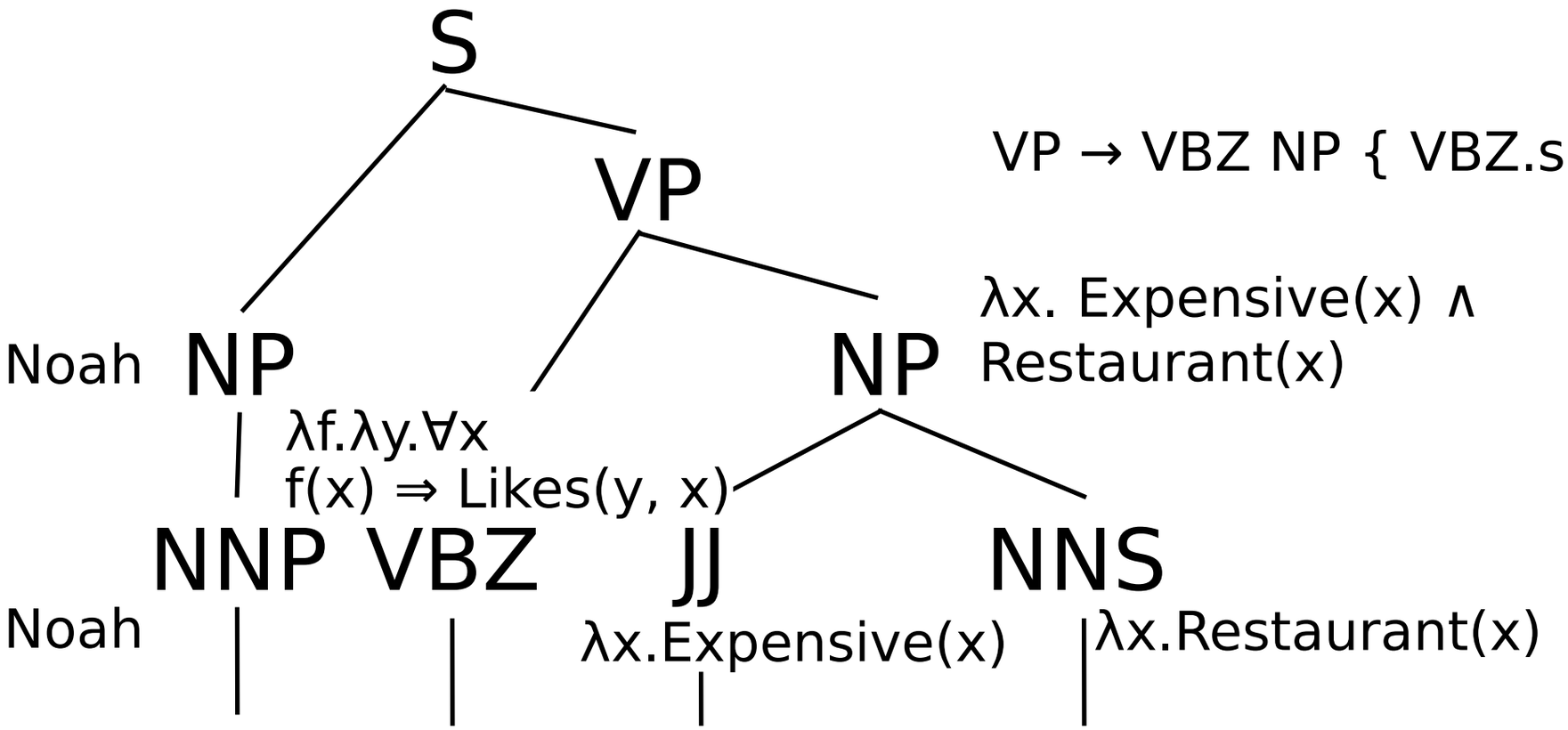
- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

# An Example




- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

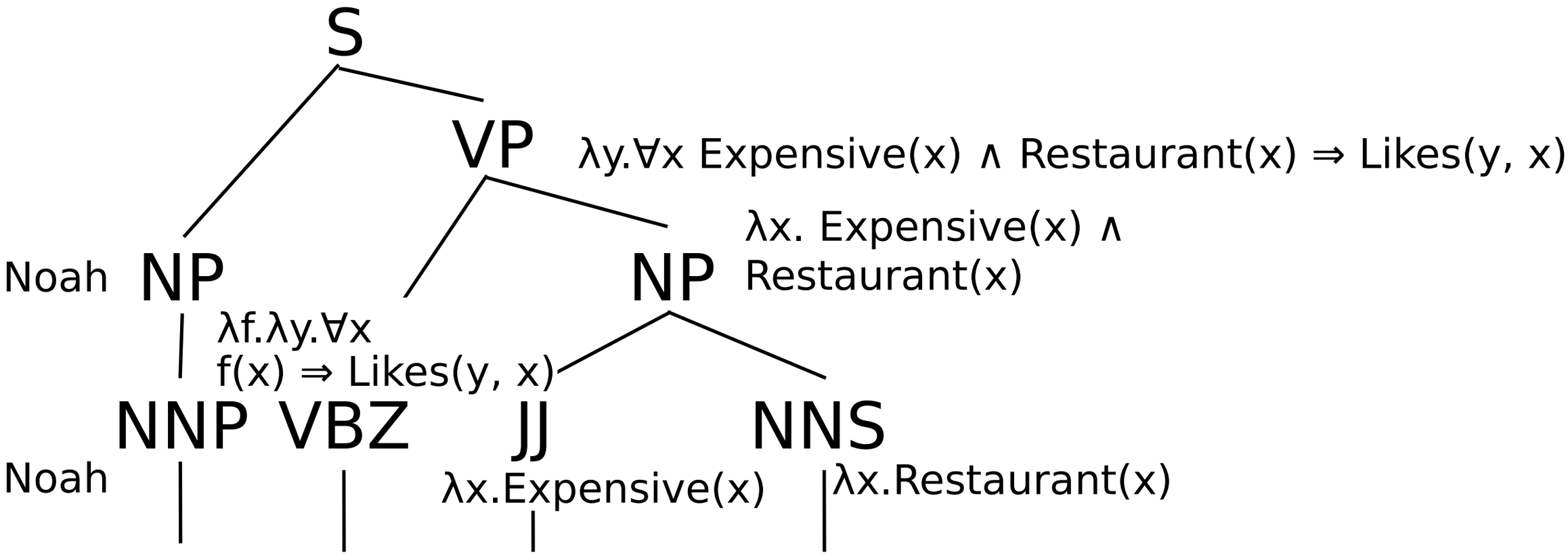
# An Example




VP → VBZ NP { VBZ.sem(NP.sem) }

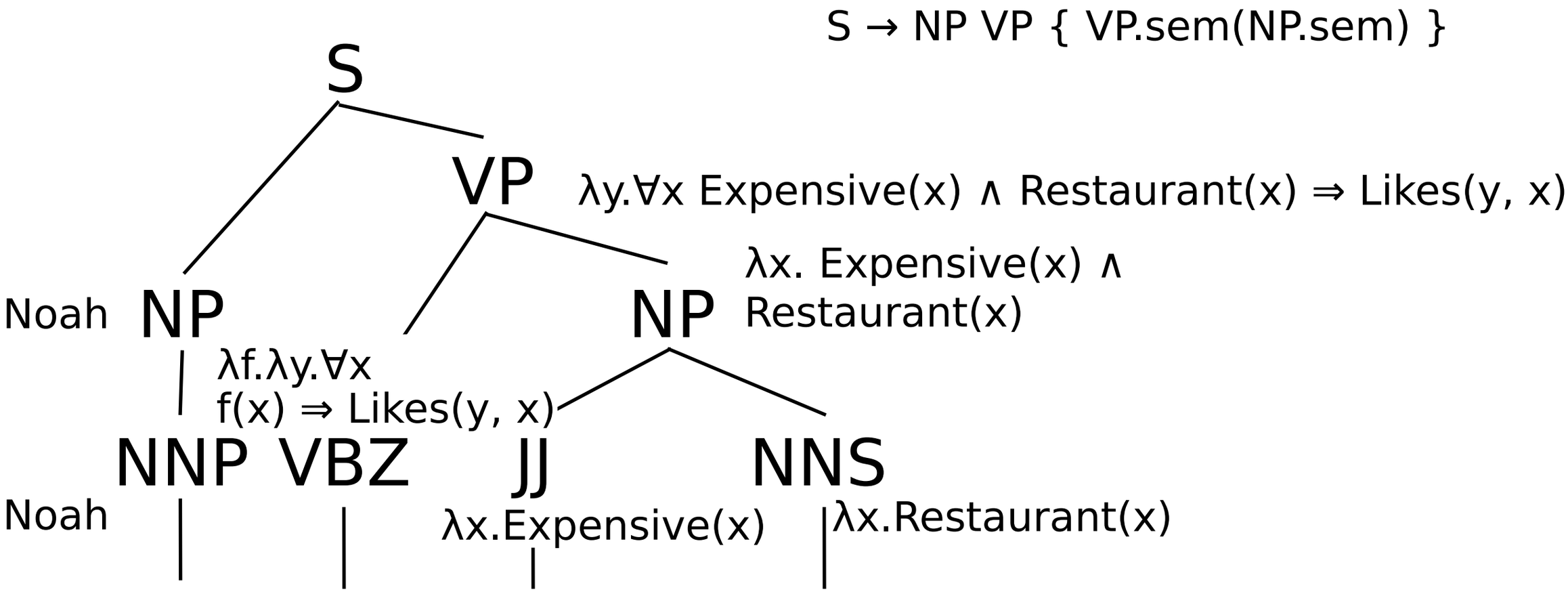
- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

# An Example



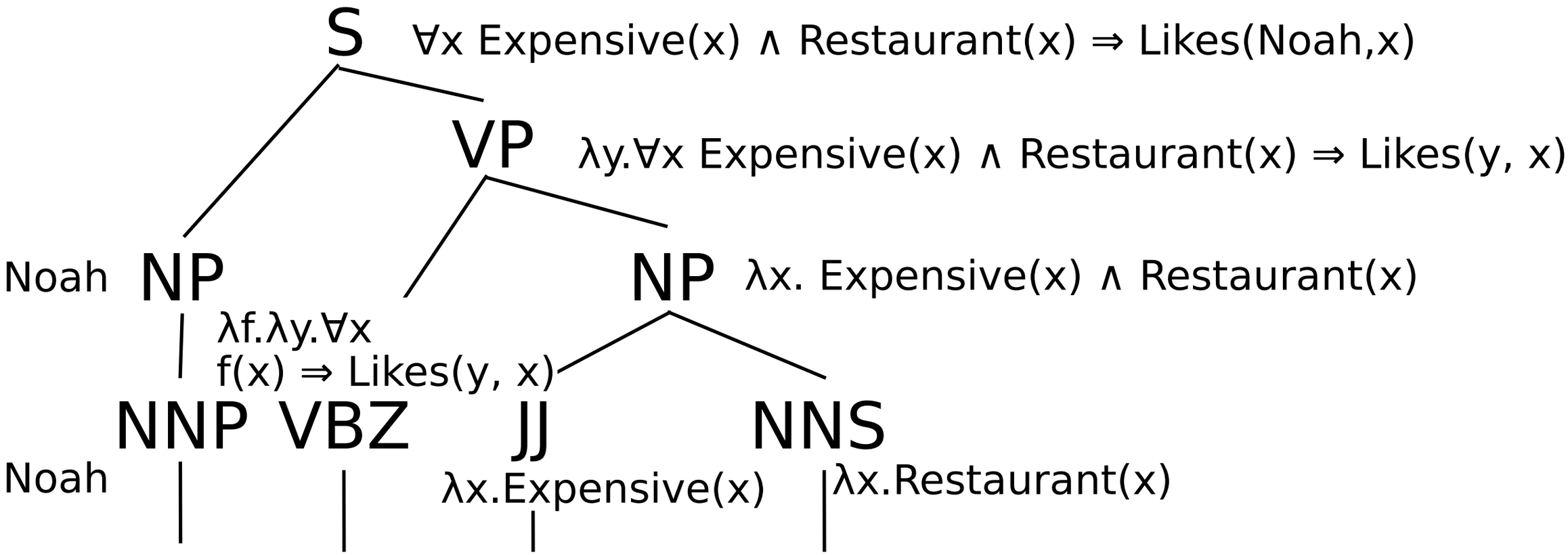

- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

# An Example



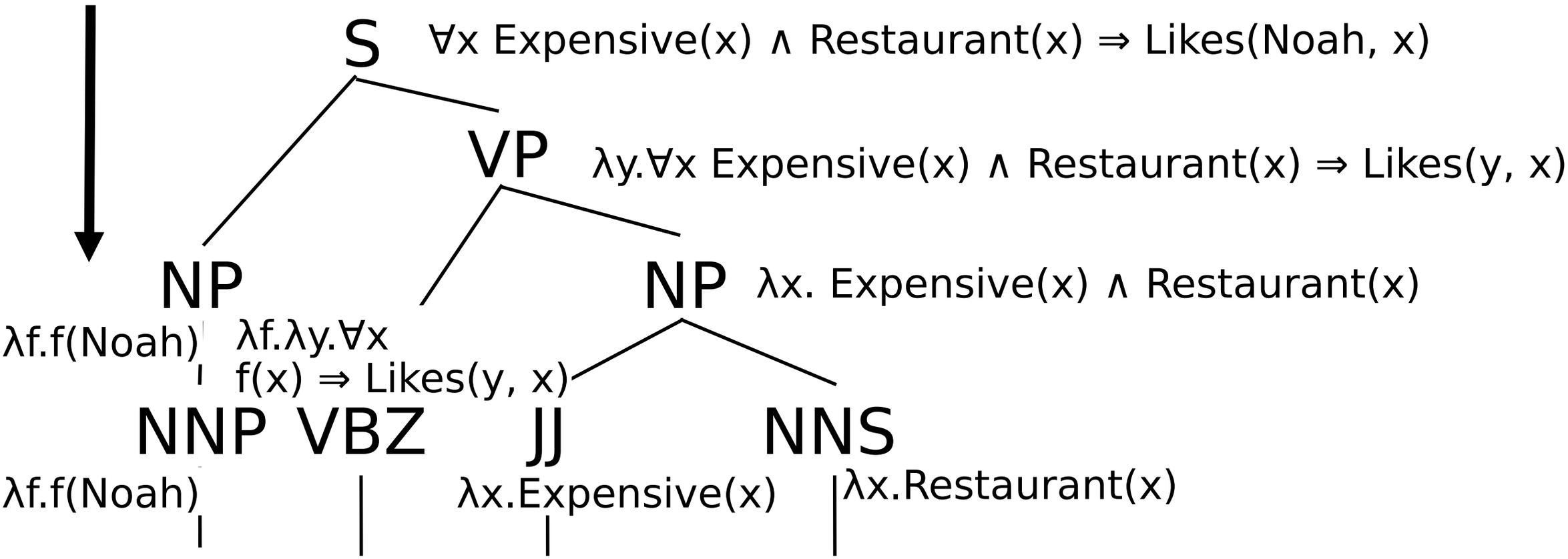

- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

# An Example




- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

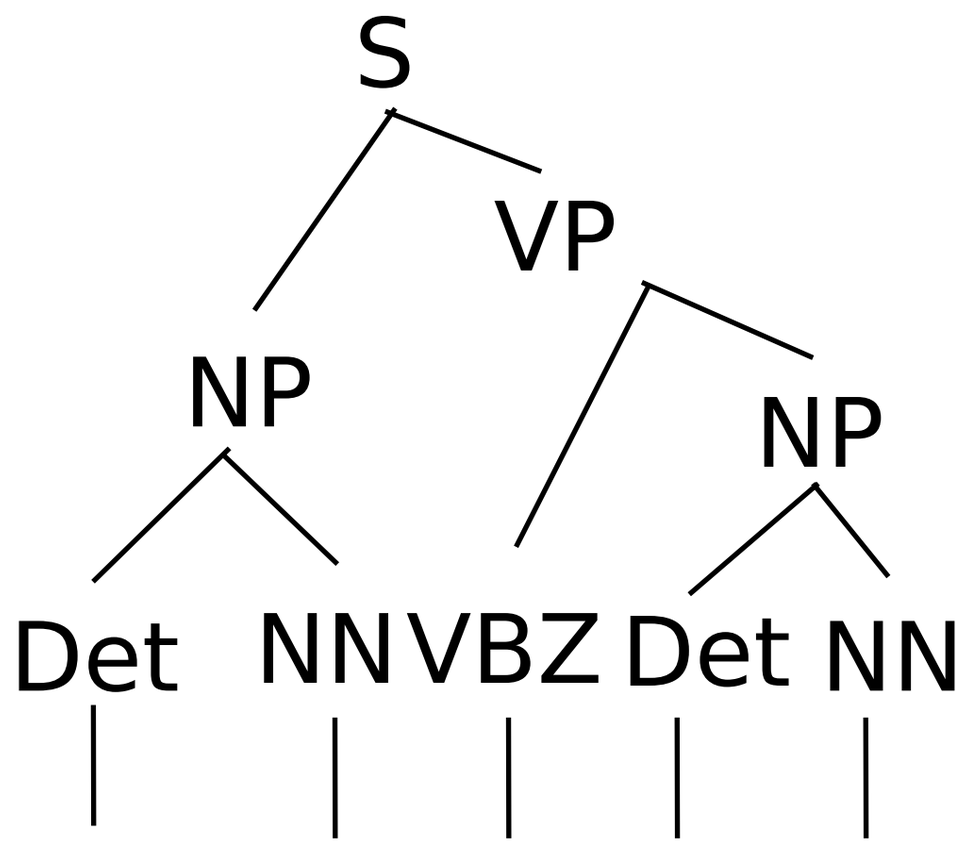
# Alternative (Following *SLP*)




- Noah likes expensive restaurants.
- ∀x Restaurant(x), Expensive(x) ⇒ Likes(Noah, x)

S → NP VP { NP.sem(VP.sem) }

# Quantifier Scope Ambiguity




S → NP VP { NP.sem(VP.sem) }

NP → Det NN { Det.sem(NN.sem) }

VP → VBZ NP { VBZ.sem(NP.sem) }

Det → every { λf.λg.∀u f(u) ⇒ g(u) }

Det → a { λm.λn.∃x m(x) ∧ n(x) }

NN → man { λv.Man(v) }

NN → woman { λy.Woman(y) }

VBZ → loves { λh.λk.h(λw. Loves(k, w)) }

- Every man loves a woman.

- ∀u Man(u) ⇒ ∃x Woman(x) ∧ Loves(u, x)
- ∃x Woman(x) ∧ ∀u Man(u) ⇒ Loves(u, x)

# This Isn’t Quite Right!

- “Every man loves a woman” really is ambiguous.
  - A seat was available for every customer
  - A toll free number was available for every customer
  - A secretary phoned up each director
  - A letter was sent to each customer
- This gives only one of the two meanings.
- One approach is to delay the quantifier processing until the end, then permit any ordering.

# Matching Syntax and Semantics

- Combinatorial Categorial Grammar (CCG)
- Five grammar rules (only)
  - Forward application A/B + B = A
  - Backward application: B + A\B = A
  - Composition: A/B + B/C = A/C
  - Conjunction: A CONJ A' = A
  - Type Raising A = X/(X\A)

# CCG Parsing

```
John = np
Mary = np
likes = (s\np)/np

Forward application
 X/Y Y => X
Backward application
 Y X\Y => X

Thus
    John    likes    Mary
     np   (s\np)/np)  np
             ---------- Forward
                s\np
        ----------- Backward
             s
```

# CCG Parsing

```
a,the np/n
old n/n
in  (np\np)/np
man,ball,park n
kicked (s\np)/np
```

```
the   old man kicked     a     ball     in      the   park
np/n n/n n  (s\np)/np np/n   n  (np\np)/np np/n  n
     -----              ------                 -----
       n                  np                     np
  ------                                 -----------
    np                                     np\np
                         -----------------
                                np
                 ----------------
                       s\np
    -------------------
             s
```

# CCG Parsing and Semantics

```
A/B:S + B:T = A:S.T
B:T + A\B:S = A:S.T

John np:j
walks (s\np):lambda X walks(X)

John                walks
np:j  s\np:lambda X walks(X)
  -----------------
      s : walks(j)

B:T + A\B:S = A:S . T
np:j + s\np:lambda X walks(X)
s : lambda X walks(X) . j
s : walks(j)
```

# CCG Parsing and Semantics

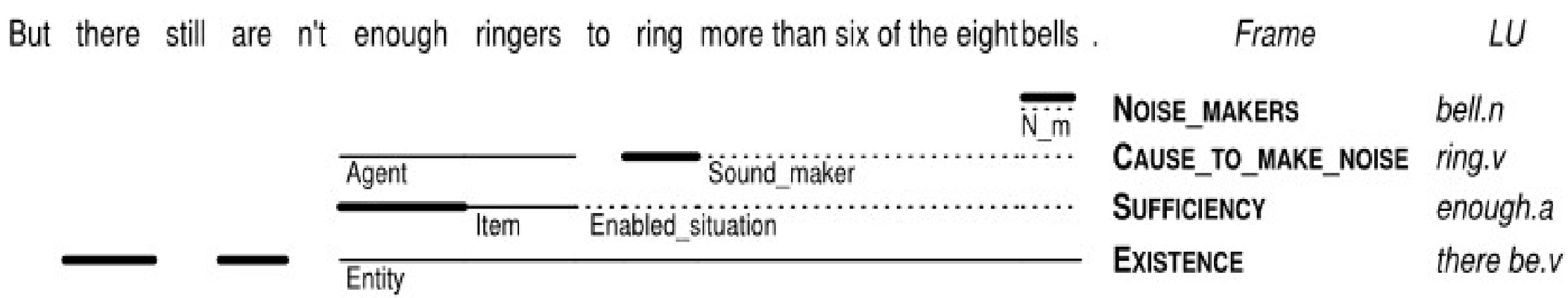
```
John np:j
Mary np:m
likes (s\np)/np: lambda Y lambda X likes(X,Y)

John                    likes                      Mary
np:j  (s\np)/np: lambda Y lambda X likes(X,Y)       m
                         ---------------------------
                        s\np: lambda X likes(X,m)
  -----------------------------
         s likes(j,m)

lambda Y lambda X likes(X,Y) . m
lambda X likes(X,m)

lambda X likes(X,m) . j
likes(j,m)
```

# Probabilistic CCGs

- Derive lexical entries from data
  - Find which entries allow parsing (constrained)
- From data with logical forms
  - Find out possible parses that derived those forms
  - But needs sentence → logical form training data
- But can work on targeted domains

# SEMAFOR

- Semantic parser (Das et al 2014)
- Uses FrameNET to identify frames
  - Fills in roles for a sentence



# Other Semantic Parsers

- Event and Entity extraction
- Semantic Role Labelling (sort of semantics)
- Intent and Slot classifiers
  - More superficial – but
  - Finds canonical meaning from sentences