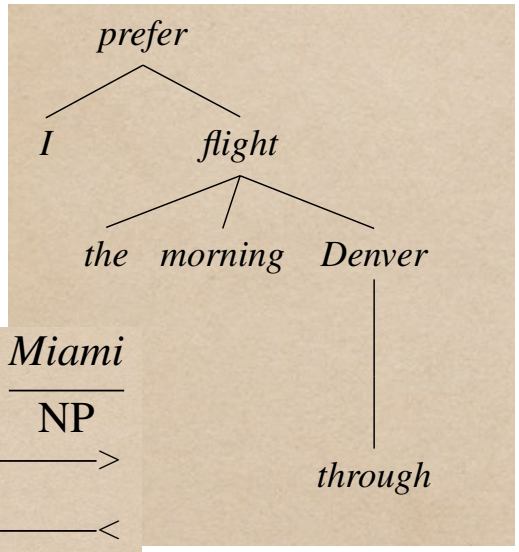
ECS763P
Natural Language Processing

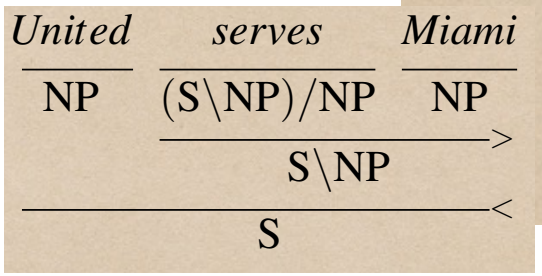
Week 9
Semantics & Applications

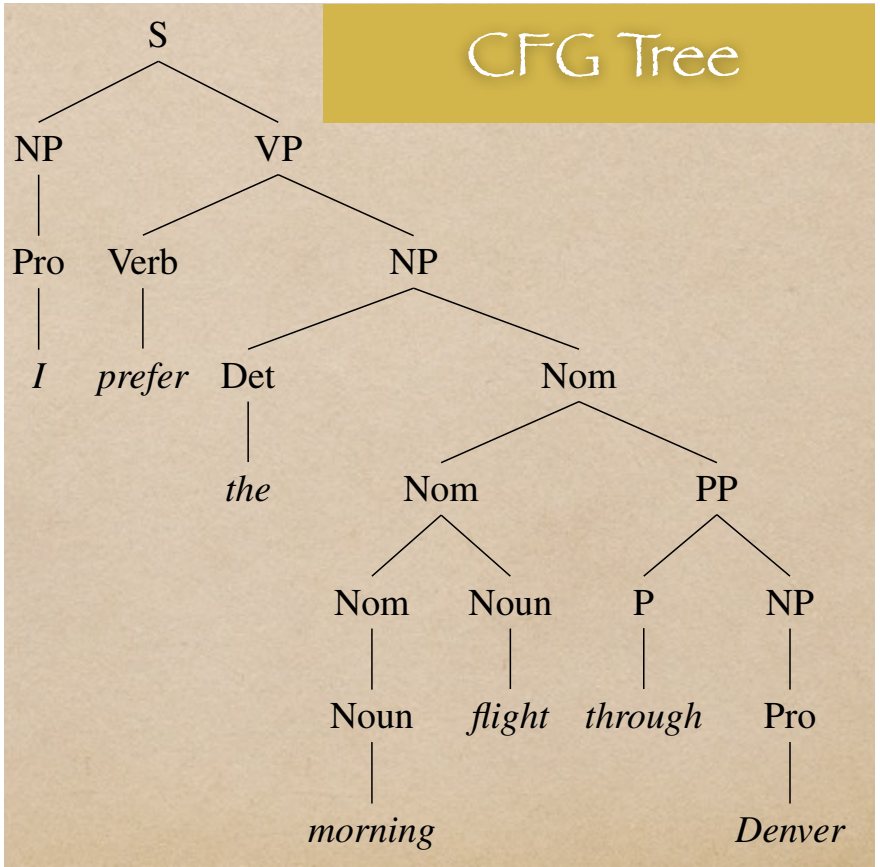
Matthew Purver
with material from Jurafsky & Martin:
Chapters 17, 18 (semantics)
Chapters 22, 23 (or 21, 28 in ed. 3 draft) (IE & QA)

Syntax

- We now know how to parse sentences
  - Sequences of words → syntactic trees
  - (or derivation trees, dependency trees etc)

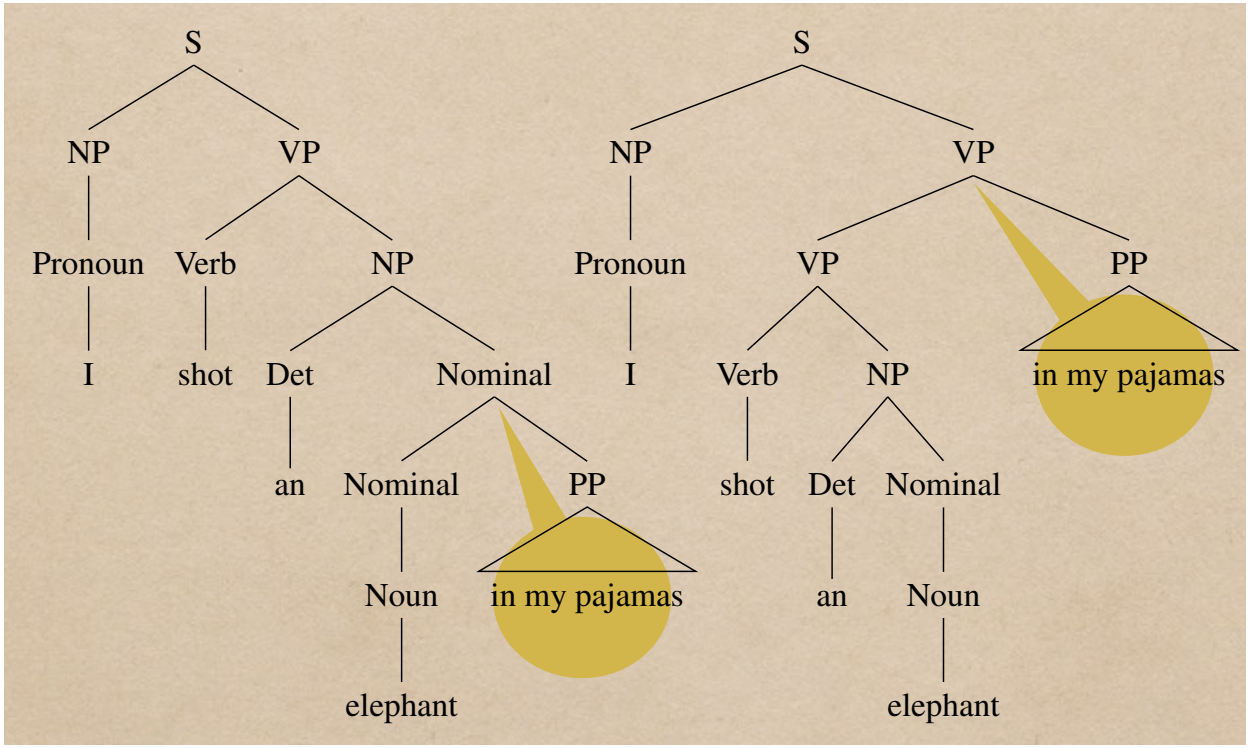






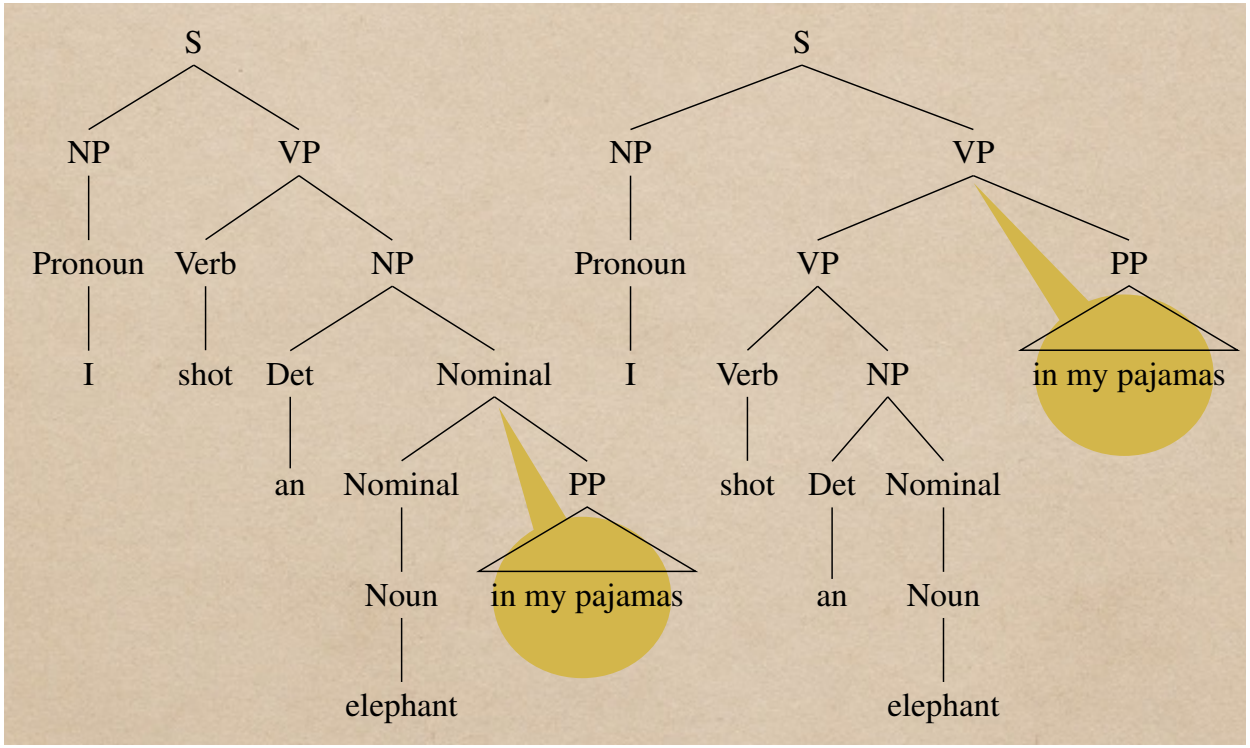
- Why do we bother?

Syntax vs Semantics



- Question-answering:
  - What did you shoot?
  - What were you wearing?
  - What was the elephant wearing?

Syntax vs Semantics



"I fail to see any interest in syntax except as a preliminary to semantics."

(Montague 1970, p223)

Syntax on its Own

- Actually, syntactic information can be very useful on its own …
  - Machine translation
    - Disambiguating alternatives
    - Subtree-based models
  - (Aspect-based) Sentiment analysis
    - Features for classification
  - Language modelling
    - Long-distance dependencies
  - Author identification
    - Stylistic features
- ... but it's usually semantics that we're really interested in

Semantic Representation

- What's a suitable way to represent meaning?
  - Verifiable matches against world e.g. database
  - e.g. RDF/OWL subject-predicate-object triples?
    - (see Semi-Structured Data module)

```
[x      is-a      elephant]
[I      shot      x]
[y      is-a      pyjamas]
[x      in        y]
```

- **Logical Forms (LFs)**
  - many possible representation languages
    - (description logics, semantic frames, discourse representations, …)
  - we'll use First-Order Logic:

$$\exists x, y.shot(I, x) \wedge elephant(x) \wedge pyjamas(y) \wedge in(x, y)$$

$$\exists x, y.shot(I, x) \wedge elephant(x) \wedge pyjamas(y) \wedge in(I, y)$$

First-Order Logic

- Atomic formulae – predicates & terms:

$$maharani$$

$$vegRestaurant(maharani)$$

$$serve(maharani, vegetables)$$

- Logical connectives:
  - conjunction "and" $\wedge$
  - disjunction "or" $\vee$
  - negation "not" $\neg$
  - implication "if … then" $\rightarrow$

$$restaurant(maharani) \wedge open(maharani)$$

$$open(maharani) \vee \neg open(maharani)$$

$$restaurant(maharani) \rightarrow serve(maharani, food$$

- Quantifiers and <u>bound</u> variables
  - existential "there exists some *x*" $\exists x$
  - universal "for all *x*" $\forall x$
  - ("bound" = only have value within the scope of the quantifier)
    - these are equivalent:

$$\exists x.restaurant(x) \wedge open(x)$$

$$\exists y.restaurant(y) \wedge open(y)$$

"Some restaurant is open"

$$\exists x.restaurant(x) \wedge open(x)$$

"All vegetarian restaurants serve vegetables"

$$\forall x.vegRestaurant(x) \rightarrow serves(x, vegetables)$$

First-Order Logic

- Translate into English:

"Jill sleeps/is sleeping/is asleep" $sleep(jill)$

"Jill is not sleeping" $\neg sleep(jill)$

"If/When Jill is tired she sleeps" $tired(jill) \rightarrow sleep(jill)$

"If Jill is tired, either she sleeps or she's irritable" $tired(jill) \rightarrow (sleep(jill) \vee irritable(jill))$

"Someone's asleep" $\exists x.sleep(x)$

"Everyone's tired and irritable" $\forall z.tired(z) \wedge irritable(z)$

"If you're tired, you're irritable" $\forall y.tired(y) \rightarrow irritable(y)$

- Translate into FOL:

"Nobody is sleeping" $\forall x.\neg sleep(x)$

"Tired people sleep" $\forall x.tired(x) \rightarrow sleep(x)$

"If Jill's sleeping and someone wakes her, she'll be angry with them" $\forall x.(sleep(jill) \wedge wake(x, jill)) \rightarrow angry(jill, x)$

Compositional Semantics

- We need some way to build an overall sentence LF from its words and its structure:
  - **Compositionality**
  - *"the meaning of a complex expression is a function of the meanings of its constituent expressions"*
  - *"[…] determined by the meanings of its constituent expressions and the rules used to combine them"*

(not actually Frege, 1890s)

- The **lambda calculus**
  - Add a new variable-binding operator *λ* to FOL
  - **Lambda abstraction** creates an anonymous function: *λx.F*
  - **Application** applies a function to an argument: *λx.F (a)*
  - **Beta reduction** rewrites application: *λx.F (a) → F[x := a]*
    - *"F with x replaced by a"*

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

$$\lambda x.restaurant(x) \wedge open(x)\ (maharani) \quad \mapsto \quad restaurant(maharani) \wedge open(maharani)$$

Typed Lambda Calculus

- We can give every FOL expression a **type**
  - *e* for entity
  - *t* for truth value (true/false)

$$maharani : \boldsymbol{e}$$

$$vegRestaurant(maharani) : \boldsymbol{t}$$

$$\exists x.restaurant(x) \wedge open(x) : \boldsymbol{t}$$

- Lambda expressions have **complex** types
  - Functions from type t_1 to type t_2

```
int doSomething(float x) { …}
```

$$\lambda x.vegRestaurant(x) : \langle e,t \rangle$$

$$\lambda S.vegRestaurant(x) \wedge S : \langle t,t \rangle$$

$$P : \langle t_1,t_2 \rangle \qquad Q : t_1 \qquad P(Q) : t_2$$

- No multiple arguments: instead, functions **to** functions

```
int doSomething(float x, float y) { …}
```

$$\lambda x.\lambda y.serves(y,x) : \langle e,\langle e,t \rangle \rangle$$

- Higher-order lambda expressions
  - Functions **from** functions

$$\lambda P.P(x) : \langle \langle e,t \rangle,t \rangle$$

$$\lambda P.\lambda Q.P(x) \wedge Q(x) : \langle \langle e,t \rangle,\langle \langle e,t \rangle,t \rangle \rangle$$

Compositionality

- Now we can build something! $P:\langle t_1,t_2\rangle$ $Q:t_1$ $P(Q):t_2$
- "Maharani serves vegetables"

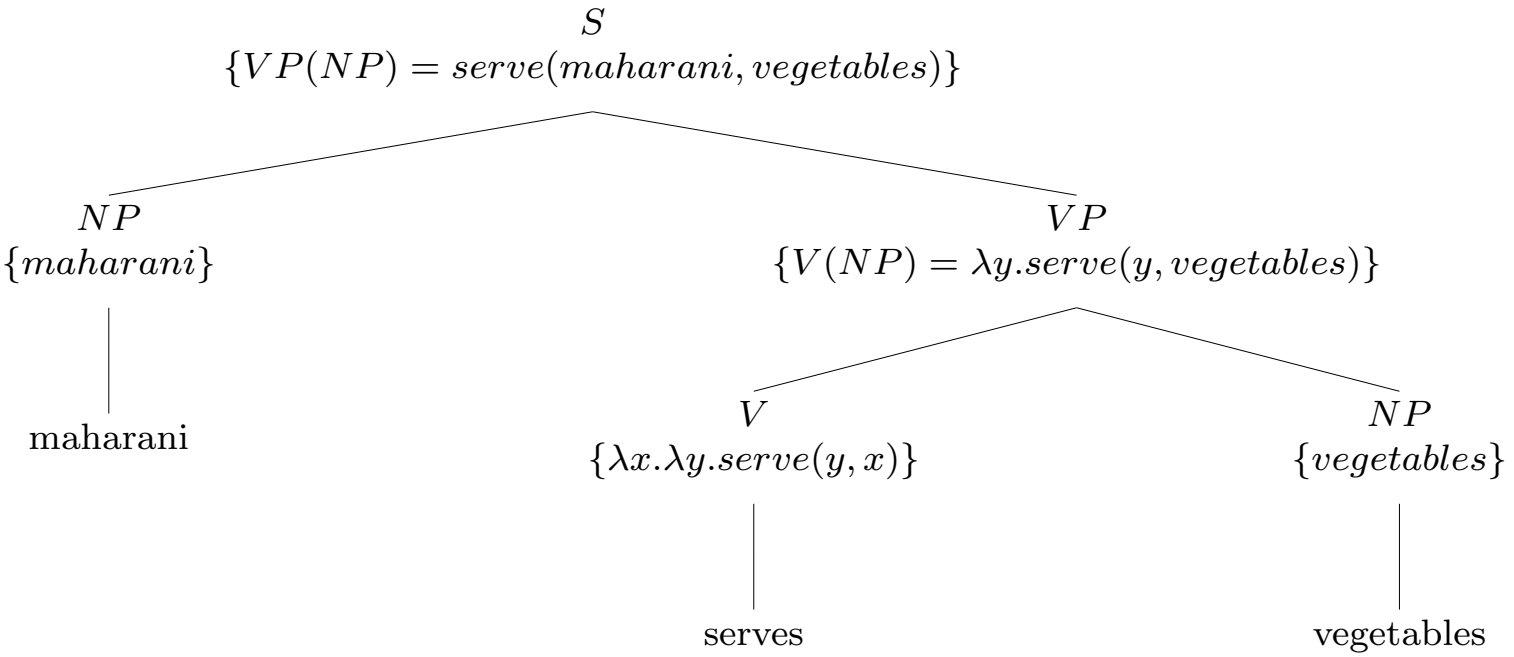
maharani	$maharani:e$
serves	$\lambda x.\lambda y.serves(y,x):\langle e,\langle e,t\rangle\rangle$
vegetables	$vegetables:e$
serves vegetables	$\lambda y.serves(y,vegetables):\langle e,t\rangle$
maharani serves vegetables	$serves(maharani,vegetables):t$

- How did we know to combine "serves" and "vegetables" first?
  - And to apply "serves" to "vegetables", not vice versa?
  - Syntax!

Syntax → Semantics

- Now we just need to add semantic rules to our grammar
  - **Semantic parsing**

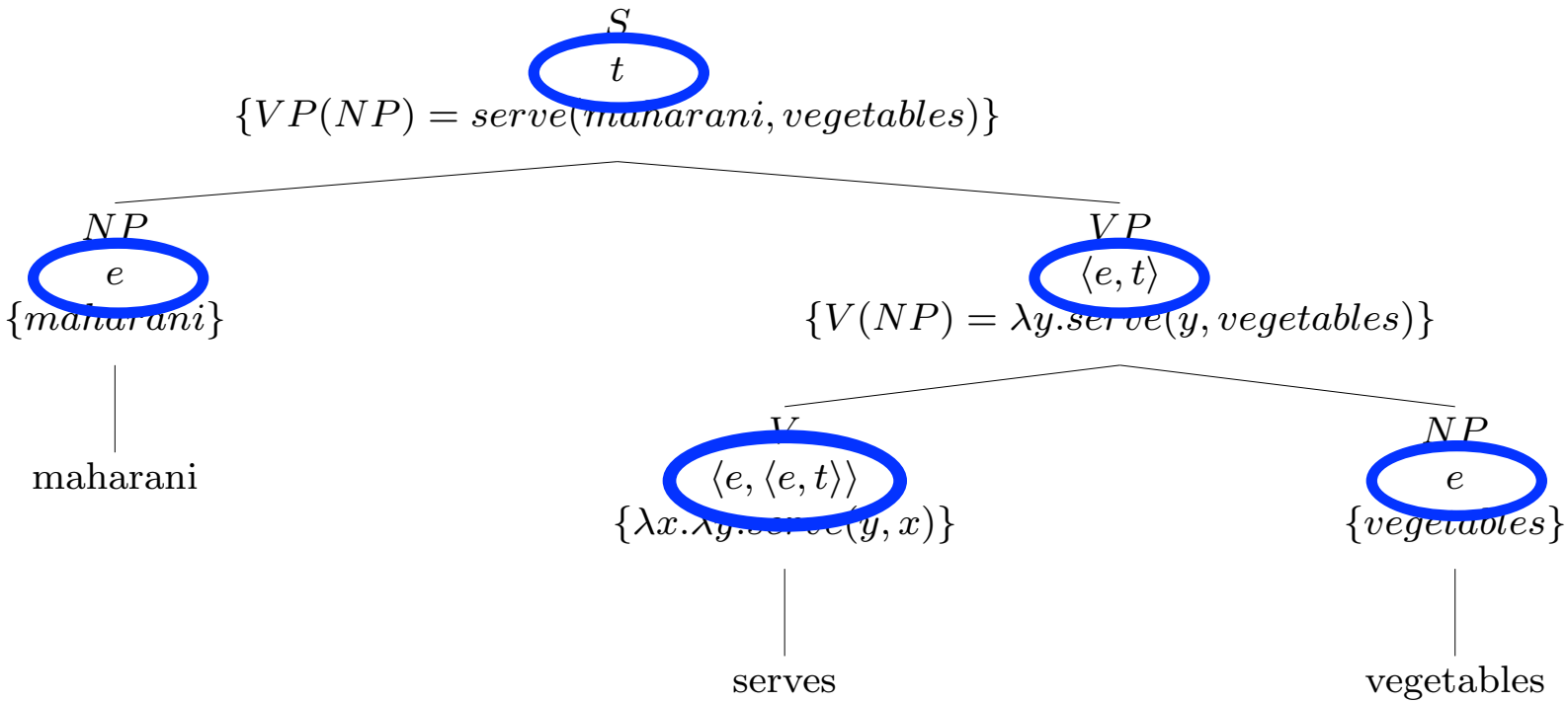
    S → NP VP {*VP(NP)*}
    VP → V NP {*V(NP)*}
    NP → "maharani" {*maharani*}



Syntax → Semantics

- Now we just need to add semantic rules to our grammar
  - **Semantic parsing**

S → NP VP {*VP(NP)*}
VP → V NP {*V(NP)*}
NP → "maharani" {*maharani*}

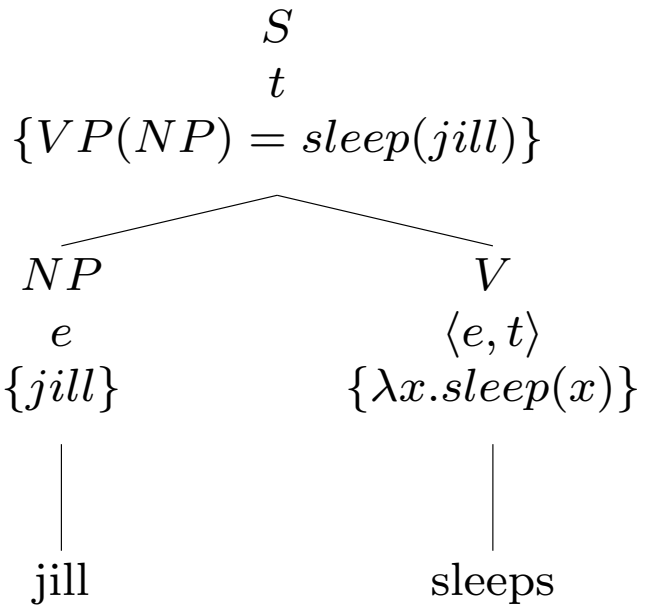


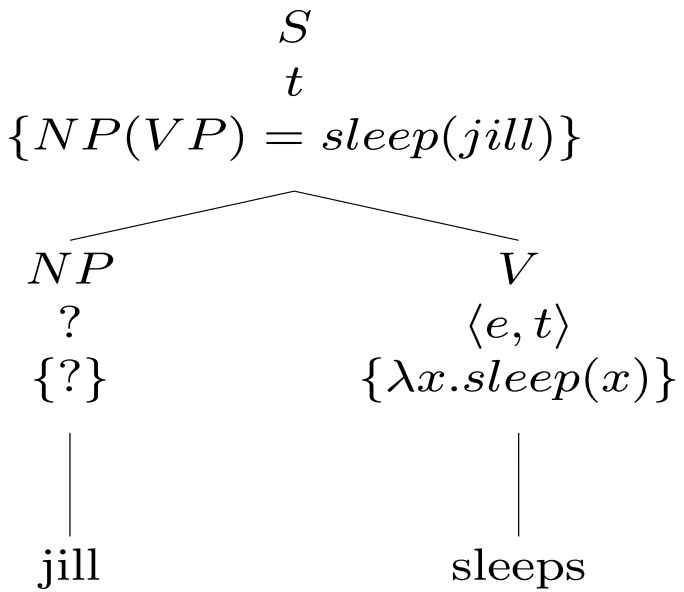
Syntax → Semantics

- What if we changed the rules?

S → NP VP ~~{*VP(NP)*}~~ {*NP(VP)*}

VP → V NP {*V(NP)*}



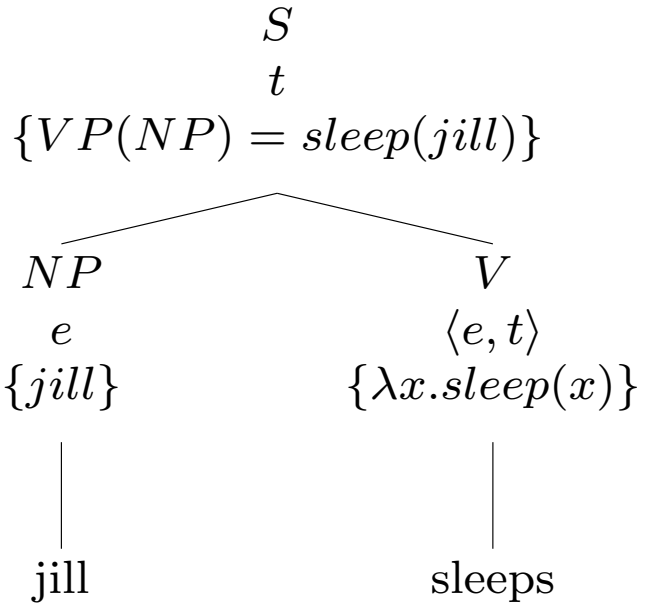


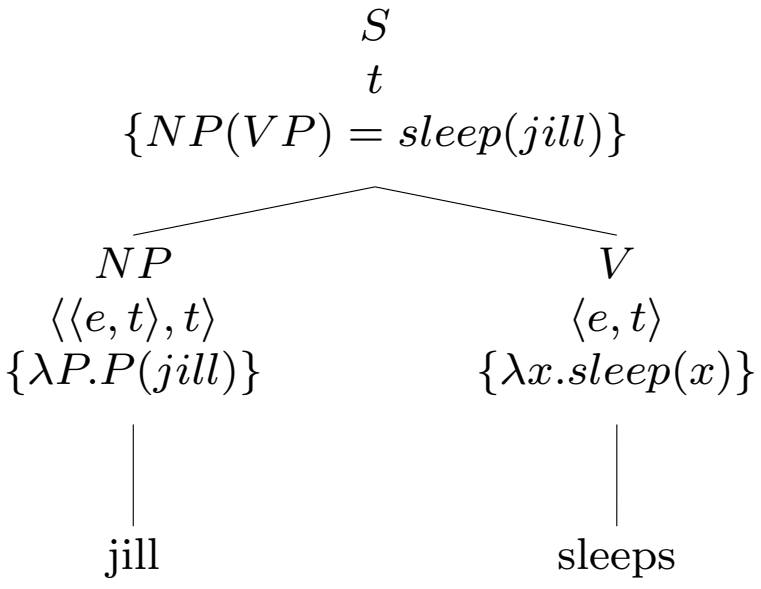
Syntax → Semantics

- What if we changed the rules?

S → NP VP ~~{*VP(NP)*}~~ {*NP(VP)*}

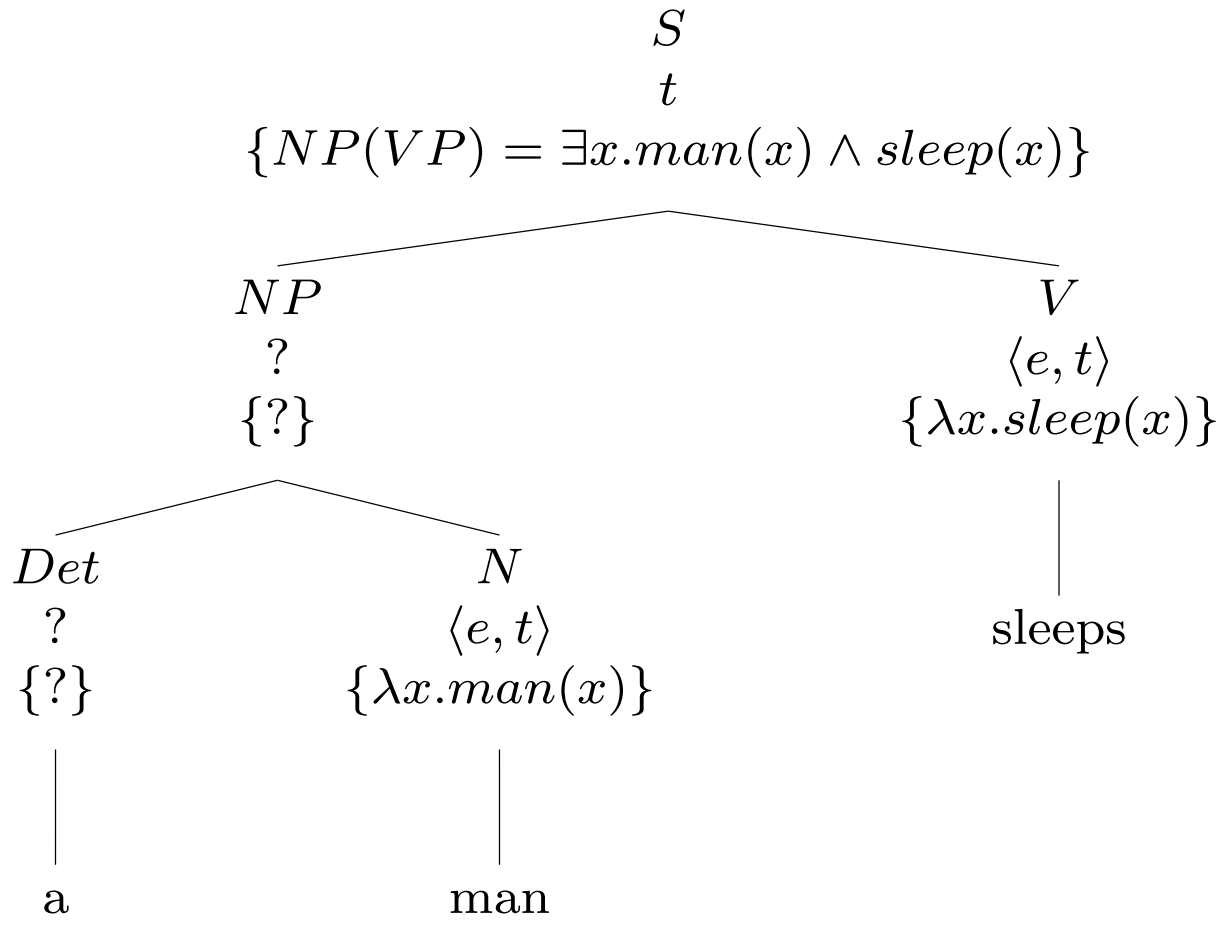
VP → V NP {*V(NP)*}





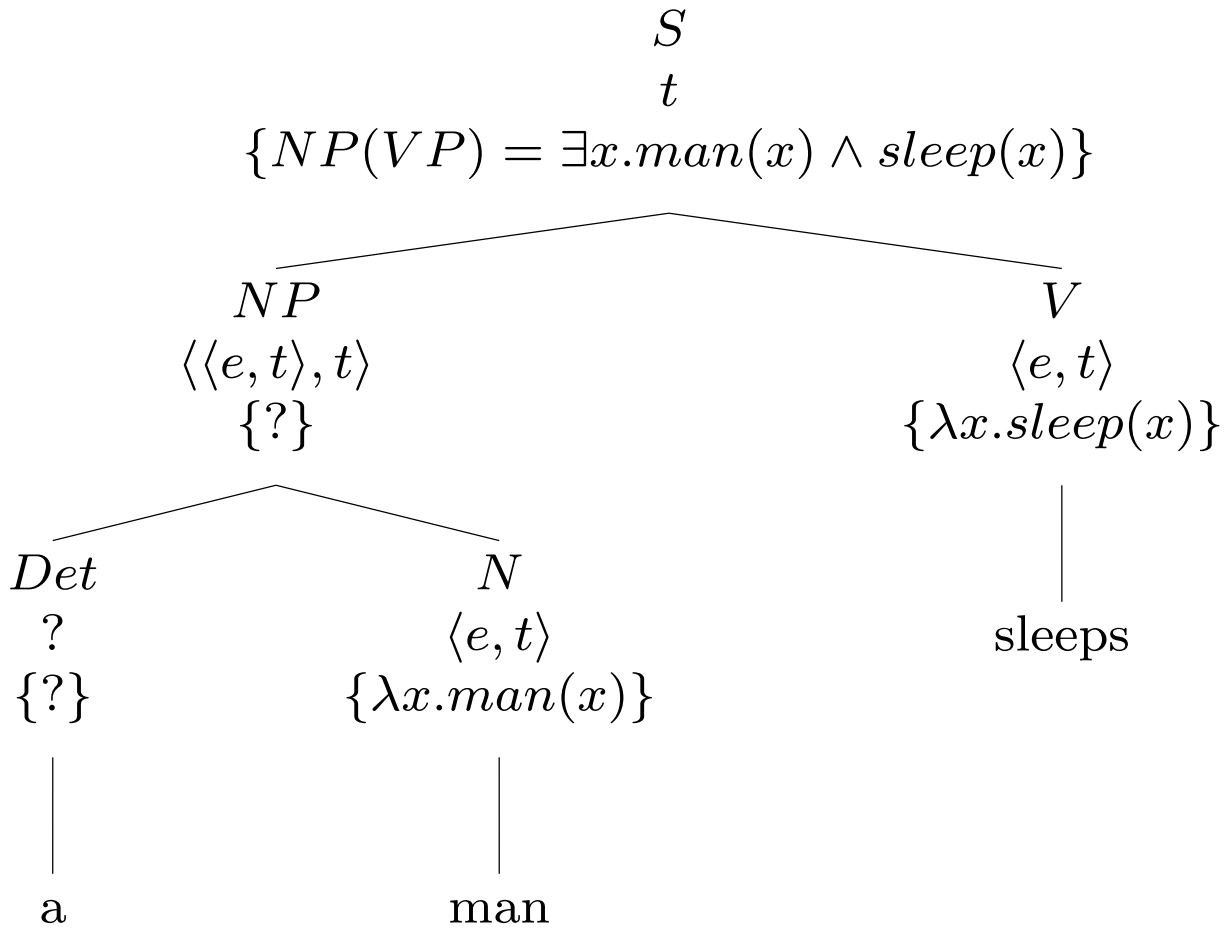
Syntax → Semantics

- What about …



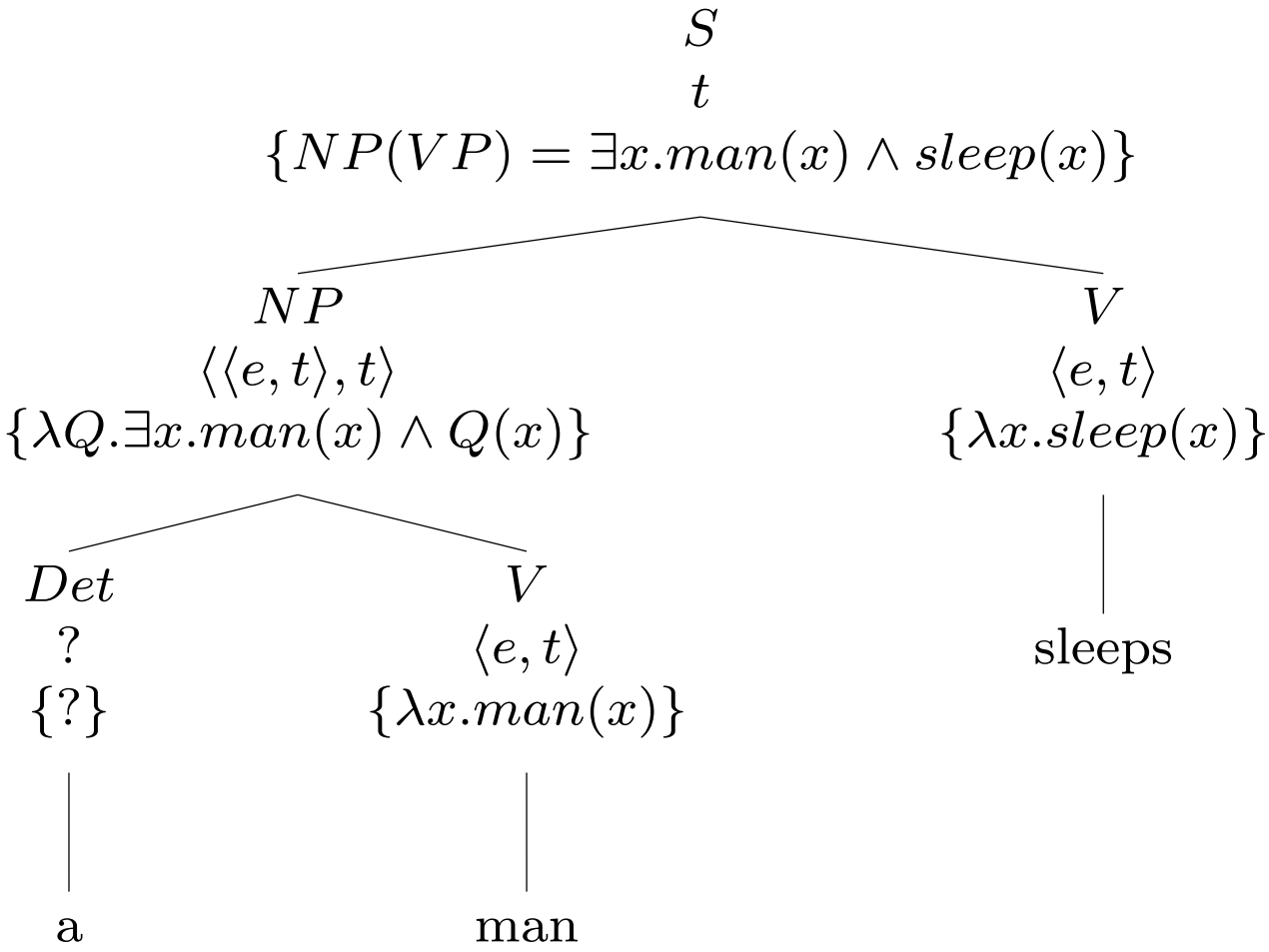
Syntax → Semantics

- What about …



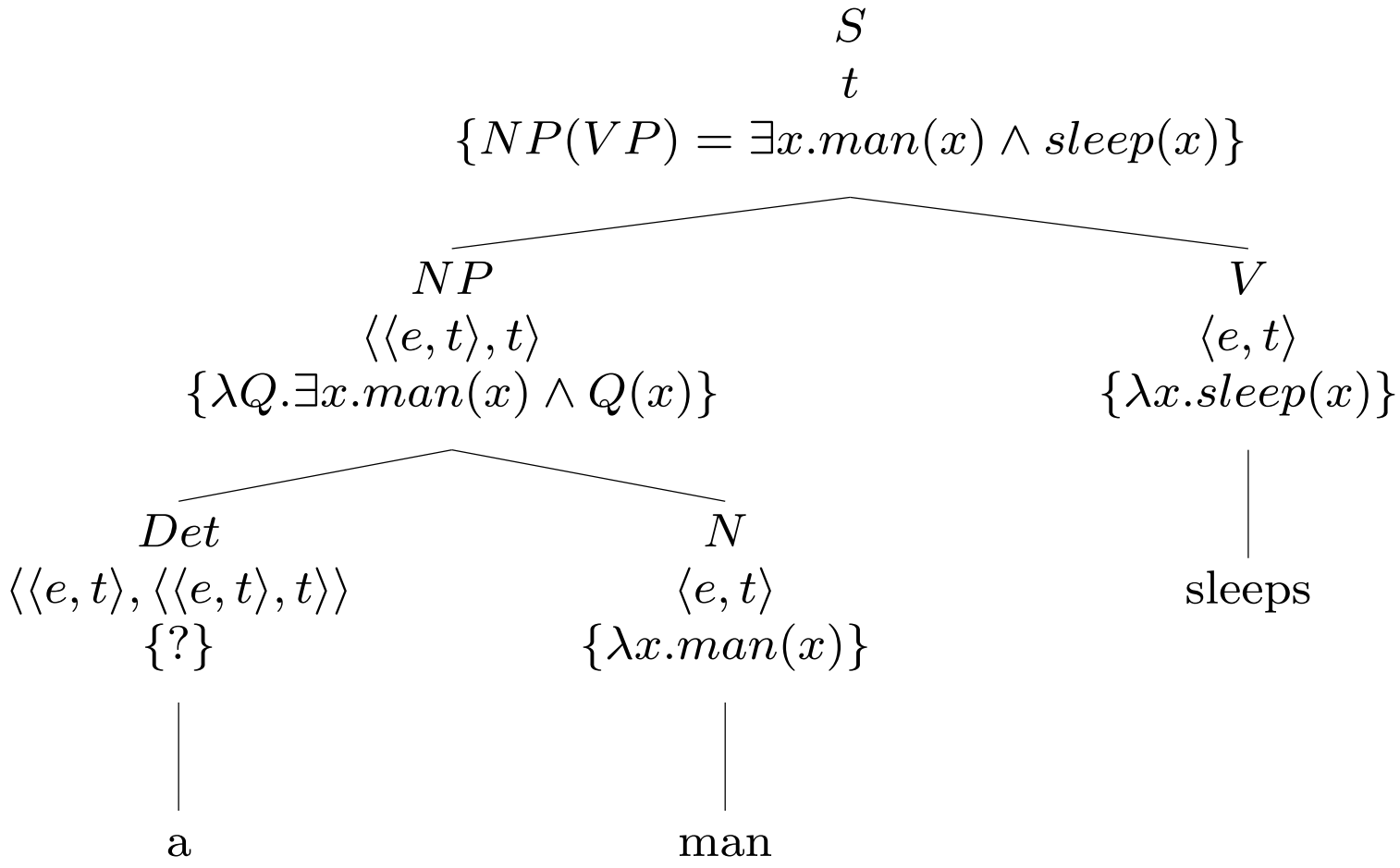
Syntax → Semantics

- What about …



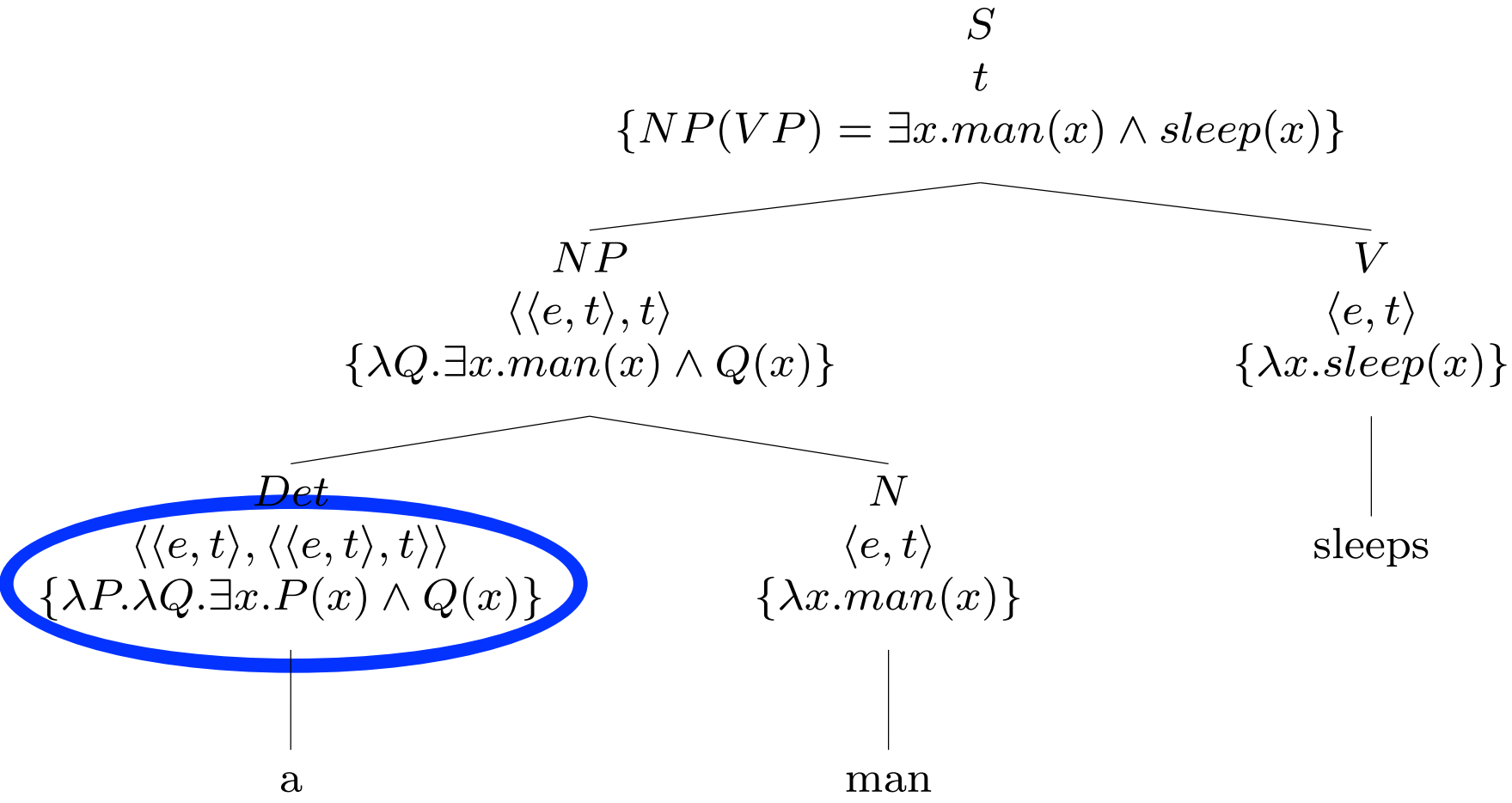
Syntax → Semantics

- What about …



Syntax → Semantics

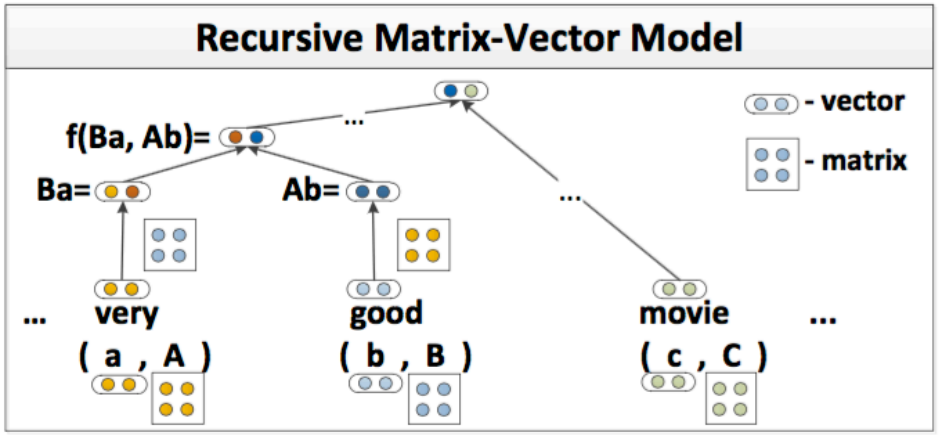
- What about …



Semantic Parsing

- This is “Montague grammar” (“PTQ”)
  - Not actually used in this simple form
  - But still the basic insight of semantic parsers
    - Add semantic rules to a grammar
    - Syntactic composition → semantic composition
    - Derive LFs for sentences
- Needs probabilistic (generative/discriminative) disambiguation
  - Can include semantic features e.g. selectional preferences, word classes
- Context-free grammars not ideal
  - CCG (combinatory categorial grammar): e.g. OpenNLP (Baldridge et al.)
  - Dependency parsers (e.g. Oepen et al, 2015))
  - Neural networks (e.g. Socher et al, 2012)

openccg.sourceforge.net



Actually …

- This kind of LF representation isn't very good …
  - What about *eat* vs *eat(cheese)* vs *eat(cheese,for lunch)* vs *eat(cheese,for lunch,slowly,on tuesday)*
  - What about tense, aspect, modifiers?
  - We often use **event terms** ("neo-Davidsonian" LFs)

$$\exists e.eat(e) \wedge subj(e, jim) \wedge obj(e, lunch) \wedge slow(e) \wedge on(e, tuesday)$$

- Even worse, FOL itself isn't very good …
  - In particular, it's terrible for things like pronoun resolution
  - We often use **discourse representation structures**

e,j,l,t	
eat(e)	lunch(l)
subj(e,j)	obj(e,l)
jim(j)	slow(e)

- But we're not going to worry about that now.

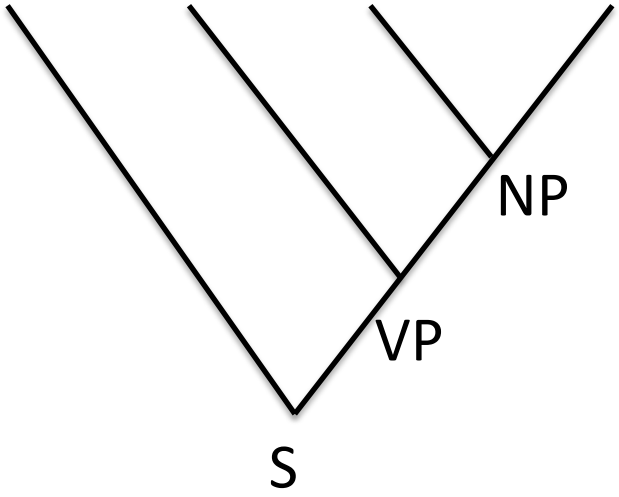
Some Basic Aspects of Language

words: mary hires a detective — TOKENISATION

parts of speech: PN VBZ DET CN — TAGGING

lemmata: mary hire a detective — STEMMING

syntax: — PARSING



semantics: $\exists$ x.detective(x) & hire(mary,x) — SEMANTIC PARSING

e,x	
hire(e)	detective(x)
subj(e,mary)	obj(e,x)

Applications

- Now, finally, we can go from words to meaning!
- So perhaps we could do a real task like:
  - Information Extraction

Citing high fuel prices, United Airlines said Friday it has increased fares by $6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes where it competes against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

FARE-RAISE ATTEMPT:	LEAD AIRLINE:	UNITED AIRLINES
	AMOUNT:	$6
	EFFECTIVE DATE:	2006-10-26
	FOLLOWER:	AMERICAN AIRLINES

  - Question-Answering

Question	Answer
Where is the Louvre Museum located?	in Paris, France
What's the abbreviation for limited partnership?	L.P.
What are the names of Odin's ravens?	Huginn and Muninn
What currency is used in China?	the yuan
What kind of nuts are used in marzipan?	almonds
What instrument does Max Roach play?	drums
What's the official language of Algeria?	Arabic
How many pounds are there in a stone?	14

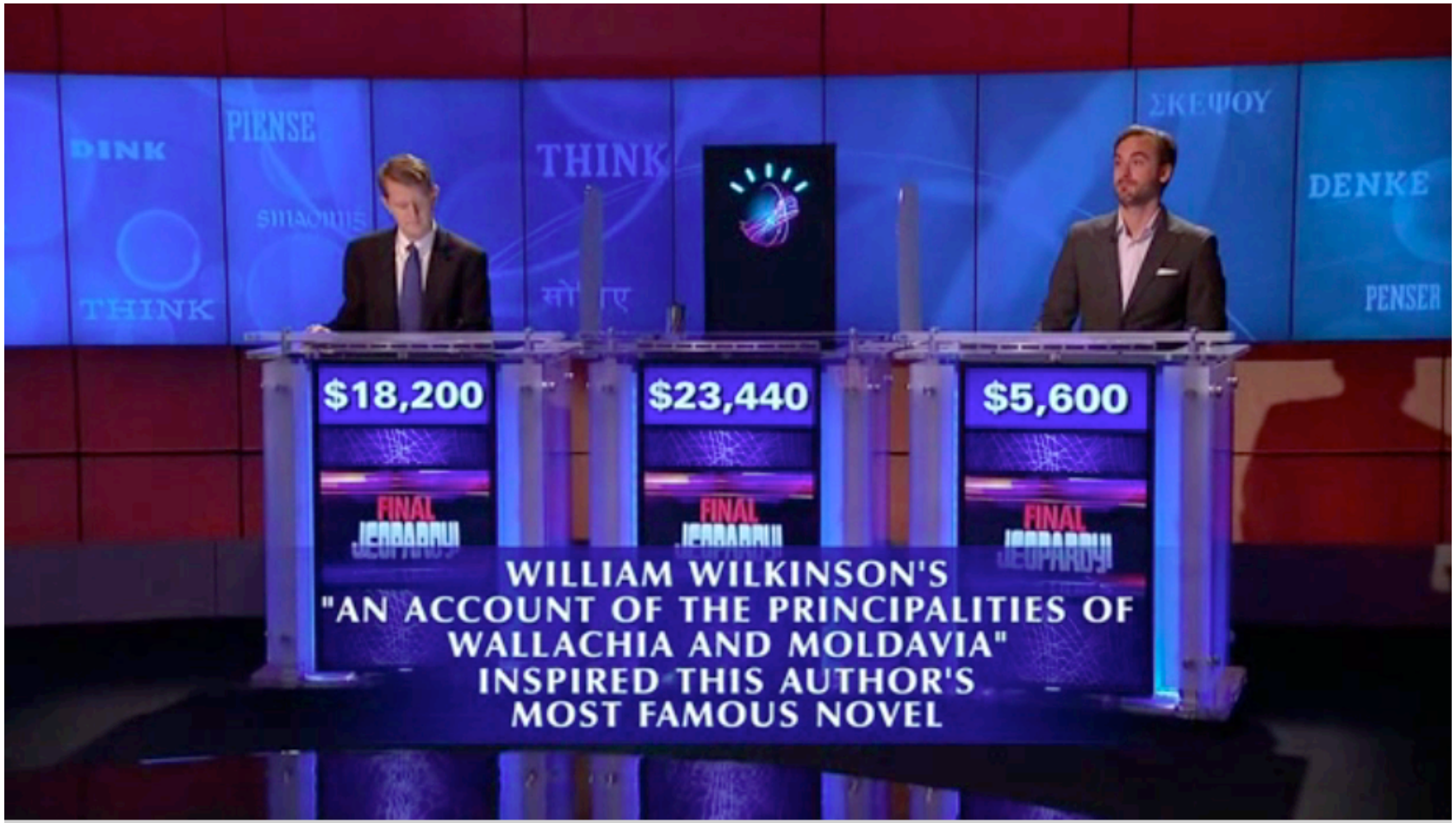
Computer finishes off human opponents on 'Jeopardy!'

By **Jason Hanna**, CNN
February 17, 2011 -- Updated 1050 GMT (1850 HKT) | Filed under: Innovation

"Jeopardy!" champs Ken Jennings, left, and Brad Rutter match wits with IBM's computer Watson on the popular game show.

cnn.com

IBM’s Watson



gizmodo.com

Information Extraction

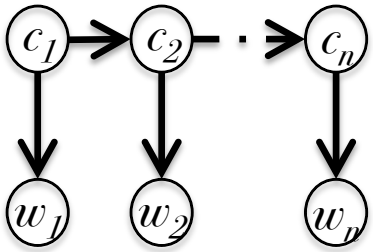
- Turn unstructured text into structured information
  - Usually to populate a database
  - Usually given a set of standard templates
- This means we're looking for entities & events, and relations between them
  - i.e. semantic parser output!
  - But first we need to identify the entities & events themselves ...

- Pipeline of standard sub-tasks:
  - Named Entity Recognition
  - Temporal Expression Normalisation
  - Event Extraction
  - Relation Detection & Extraction
  - Template-Filling

Citing high fuel prices, [$_{ORG}$ **United Airlines**] said [$_{TIME}$ **Friday**] it has increased fares by [$_{MONEY}$ **$6**] per round trip on flights to some cities also served by lower-cost carriers. [$_{ORG}$ **American Airlines**], a unit of [$_{ORG}$ **AMR Corp.**], immediately matched the move, spokesman [$_{PER}$ **Tim Wagner**] said. [$_{ORG}$ **United**], a unit of [$_{ORG}$ **UAL Corp.**], said the increase took effect [$_{TIME}$ **Thursday**] and applies to most routes where it competes against discount carriers, such as [$_{LOC}$ **Chicago**] to [$_{LOC}$ **Dallas**] and [$_{LOC}$ **Denver**] to [$_{LOC}$ **San Francisco**].

Named Entity Recognition

- Sequence labelling tasks
  - (we know how to do this!)
  - HMMs, CRFs, RNNs
- Need suitable tagset
  - BIO = beginning, inside, outside
  - ORG vs PER vs TIME vs LOC vs …
- Need to choose features
  - Words, (ngram) context
  - POS tags
  - Capitalisation
  - Hyphenation, punctuation
  - Dictionaries
- Remember pre-processing should be suitable
    - text normalisation, POS-tagging, lemmatisation …
- Same approach to detect & tag temporal expressions
  - Then rule-based normalisation e.g. “this weekend” → 2017/03/11-2017/03/12



Words	BIO Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O
matched	O	O
the	O	O
move	O	O
,	O	O
spokesman	O	O
Tim	B-PER	I-PER
Wagner	I-PER	I-PER
said	O	O
.	O	O

Relation & Event Extraction

- Parsing (e.g. dependency parsing) gives semantic relations, mostly verb predicate-argument (e.g. verb-subject-object) structures:
  - [United Airlines] has increased [fares]
  - [American] immediately matched [the move]
- Many relations (and some events) not expressed by verbs:
  - Spokesman [Tim Wagner]
  - [United], a unit of [UAL Corp.]
- Often use sets of patterns to find relevant relations/events and tag with relation types
  - Manually specified
  - Supervised learning
  - Distant supervision (e.g. using patterns to extract large numbers of likely candidates)
- May try to order events by time
  - Temporal expressions, discourse structure (keywords/adverbs, sentence order)
- Final stage of template-filling matches relations with slots
  - Combination of manual rules & statistical classifiers

Question-Answering

- Providing answers to questions (usually **factoid questions**):
  - What is the capital of Estonia?
  - Who founded Virgin Airlines?
  - Which is the biggest port on the Mediterranean?
  - When is the next leap year?
  - **Answer Type:** usually person, location, date, city etc.
- Compare:
  - Information Retrieval: returning relevant text passages
  - Information Extraction: finding pre-specified information

- Two families of approaches:
  - IR ("text-based") method
    - Retrieve relevant short passages; rank by answer type
  - Semantic ("knowledge-based") method
    - Parse question & answer, match semantic relations

IR-based QA

- (You may have done this in the IR module)

- Question processing:
  - Extract keywords
    - (see IR module)
    - Maybe remove wh-words, do query expansion
  - Classify expected answer type
    - e.g. supervised classification, n-gram features

- Answer retrieval:
  - Document retrieval: standard IR
  - Passage retrieval: more specific passage
    - Standard IR methods, weighted keyword relevance
    - Rank based on number & type of named entities
  - Answer extraction
    - Template/pattern-matching
    - N-gram extraction & combination ("tiling")

Semantic QA

- Parse query: "which states border Texas?" $\lambda x.state(x) \wedge border(x, texas)$
- Parse **lots** of text into a knowledge base
- Similar basic processing steps to IE:
  - Named Entity Recognition (after POS-tagging etc)
  - Temporal Expression Normalisation
  - Semantic Parsing
- Query & candidate fact processing:
  - Parse to semantic LF for question
    - Often needs specific grammars, combined with supervised machine learning for disambiguation
  - Map LF to database/triple format
    - Rule-based (manually specify patterns)
    - Supervised learning: associate parse sub-trees with triples
    - Distant supervision
      - e.g. find common patterns "X borders Y"
      - Look for "X … Y" and find new common patterns "X is bounded by Y", "X neighbours Y" etc

$state(oklahoma)$

$border(oklahoma, texas)$

$state(kansas)$

…

What about …

- QA needs to use data like Wikipedia:

Richard Branson

From Wikipedia, the free encyclopedia

Sir Richard Charles Nicholas Branson (born 18 July 1950) is an English business magnate, investor and philanthropist.[4] He founded the Virgin Group, which controls more than 400 companies.[5]

Branson expressed his desire to become an entrepreneur at a young age. At the age of sixteen his first business venture was a magazine called *Student*.[6] In 1970, he set up a mail-order record business. In 1972, he opened a chain of record stores, Virgin Records, later known as Virgin Megastores. Branson's Virgin brand grew rapidly during the 1980s, as he set up Virgin Atlantic airline and expanded the Virgin Records music label.

- Who founded the Virgin Group?
- When did Richard Branson set up Virgin Atlantic?
- What was Branson's first business venture?

And what about …

- Did Branson enter the Australian aviation market? When?

The Sydney Morning Herald | Business

News & Views | Markets | Quotes | Workplace | Consumer | Property | Innovation

The year was 2001, one year after the legendarily brash, anti-establishment renegade had pinpointed an opportunity to enter the Australian aviation market. Branson did so against the backdrop of a corporate landscape littered with the carcasses of other airline aspirants such as Compass and Impulse, all of whom had tried and failed to get a foothold in a duopoly market locked up the then government-owned Qantas and the Singapore Airlines-Air New Zealand-controlled Ansett.

Summary

- We want to extract meaning from text
  - So we need to represent meaning in a logical form
- First order logic (FOL) as a reasonable framework
- Typed lambda calculus to make FOL compositional
- Semantic parsing by augmenting syntactic parsing
  - (or dependency parsing, recursive neural nets, …)

- We can combine what we know now for a real task:
  - Information Extraction
  - Question-Answering