Semantic parsing

Lê Thanh Hương
School of Information and Communication Technology
Email: huonglt@soict.hust.edu.vn

ONE LOVE. ONE FUTURE.

Definition

- Semantic parsing is mapping from natural language to its logic representation.
- Shallow parsing: semantic role labeling
- Deep parsing: logic representation allowing inference

Chapter 17. Representing Meaning. In book Speech and Language Processing. Dan Jurafsky and James Martin. 2rd edition. Prentice Hall.

Applications

- Question-answering
- Chatbot
- Robot controlling
- Machine translation
- Text summarization

What counts as understanding?

- … if can react suitably
  - e.g: “Put all the toys in the basket”
- … if can determine right from wrong

What counts as understanding?

- … if can answer related question?
  - Easy: Mai eats cake. → What did Mai eat?
  - Difficult: White's first move is P-Q4. → Can black checkmate?

- … if can translate: depends on target language
  - English – English?
  - English – French? possible
  - English – logic ? need deep understanding

    - All fishes can swim
    - $\forall x$ [fish(x) $\rightarrow$ can_swim(x)]

Background on logic

3 basic object types:

1. Value - Booleans
   - semantic value of sentence
2. Entities
   - time, table, chair...
3. Function
   - return binary value (predicate), e.g frog(x), green(x)
   - Can return a function
   - Can take function as argument

Logic: Lambda terms

- Lambda terms:
  - A way of writing “anonymous functions”
    - No function header or function name
    - But defines the key thing: **behavior** of the function
    - Just as we can talk about 3 without naming it “x”
  - Let square = λp p*p
  - Equivalent to int square(p) { return p*p; }
  - Format of a lambda term:
    - λ<varible> <expression>

Logic: Lambda terms

- Lambda terms:
  - Let square = λp p*p
  - then square(3)   = (λp p*p)(3) = 3*3
  - Note: square(x) is not a function. It’s just the value of x*x.
  - But: λ**x** square(x) = λx x*x = λp p*p = square
  - Let even = λp (p mod 2 == 0)   a predicate returns True/False
  - even(x) = true if x is even

  - How about even(square(x))?
  - λx even(square(x)) = true for x with square(x) event
    - λx (even(x*x)) = λx (x*x mod 2 == 0)

Logic: Some predicate

- most – a predicate on 2 predicates on entities
  - most(pig, big) = "most pigs are big"
    - Equivalently, most(λx pig(x), λx big(x))
  - returns true if most of the things satisfying the 1st predicate also satisfy the 2nd predicate
- similarly for other quantifiers
  - all(pig,big) (equivalent to $\forall$x pig(x) $\Rightarrow$ big(x))
  - exists(pig,big) (equivalent to $\exists$x pig(x) AND big(x))

Predicate representation

- Gilly swallowed <u>a</u> goldfish
  - swallowed(Gilly, goldfish)

  goldfish isn't the name of a unique object the way Gilly is

- In particular, don't want
  - Gilly swallowed a goldfish and Milly swallowed a goldfish

to translate as

- swallowed(Gilly, goldfish) AND swallowed(Milly, goldfish)

since probably not the same goldfish …

Use quantifiers

- Gilly swallowed <u>a</u> goldfish
  - swallowed(Gilly, goldfish)
- Better: ∃g goldfish(g) AND swallowed(Gilly, g)
- Or use quantifiers
  - exists(λg goldfish(g), λg swallowed(Gilly,g))
  - Equivalently: exists(goldfish, swallowed(Gilly))
    - "In the set of goldfish there exists one swallowed by Gilly"

- Mai likes small cats.
- Mai likes the cat whose name is Tom.

Time

- Gilly swallowed a goldfish
  - Previous attempt: exists(goldfish, λg swallowed(Gilly,g))
- Improve to use tense:
  - Instead of the 2-arg predicate swallowed(Gilly,g)
    try a 3-arg version swallow(t,Gilly,g)   where t is a time
  - Now we can write:
    ∃t past(t) AND exists(goldfish, λg swallow(t,Gilly,g))
  - "There was some time in the past such that a goldfish was among the objects swallowed by Gilly at that time"

Event Properties

- Gilly swallowed a goldfish
  - Previous: ∃t past(t) AND exists(goldfish, swallow(t,Gilly))
- Why stop at time? An event has other properties:
  - [Gilly] swallowed [a goldfish] [on a dare] [in a telephone booth] [with 30 other freshmen] [after many bottles of vodka had been consumed].
  - Specifies who what why when …
- Replace time variable t with an event variable e
  - ∃e past(e), act(e,swallowing), swallower(e,Gilly), exists(goldfish, swallowee(e)), exists(booth, location(e)), …
    - As with probability notation, a comma represents AND
    - Could define past as λe ∃t before(t,now), ended-at(e,t)

Quantifier Order

- Example
  - **`In this country a woman gives birth every 15 min. Our job is to find that woman and stop her.`**
  - $\exists$woman $\big(\forall$15min gives-birth-during(woman, 15min)$\big)$
  - $\forall$15min $\big(\exists$woman gives-birth-during(15min, woman)$\big)$

Intensional Arguments

- Willy wants a unicorn
  - ∃e act(e,wanting), wanter(e,Willy), exists(unicorn, λu wantee(e,u))
    - "there is a unicorn u that Willy wants"
    - here the wantee is an individual entity
  - ∃e act(e,wanting), wanter(e,Willy), wantee(e, λu unicorn(u))
    - "Willy wants any entity u that satisfies the unicorn predicate"
    - here the wantee is a <u>type</u> of entity

- Willy wants Lilly to get married
  - ∃e present(e), act(e,wanting), wanter(e,Willy), wantee(e, λe' [act(e',marriage), marrier(e',Lilly)])
  - "Willy wants any event e' in which Lilly gets married"
  - Here the wantee is a <u>type</u> of event
  - Sentence doesn't claim that such an event exists

- Intensional verbs besides want: hope, doubt, believe,…

Nouns and Their Modifiers

- Expert λg expert(g)
- big fat expert λg big(g), fat(g), expert(g)

- Baltimore expert (white-collar expert, TV expert …)
  - λg Related(Baltimore, g), expert(g) – expert from Baltimore

- Baltimore expert (white-collar expert, TV expert …)
  - λg Related(Baltimore, g), expert(g) – expert from Baltimore
  - Or with different intonation:
    - λg (Modified-by(Baltimore, expert))(g) – expert on Baltimore
  - Can't use Related for that case: law expert and dog catcher
    = λg Related(law,g), expert(g), Related(dog, g), catcher(g)
    = dog expert and law catcher

Speech Acts

- What is the meaning of a full sentence?
  - Depends on the punctuation mark at the end.
  - Billy likes Lili. → assert(like(B,L))
  - Billy likes Lili? → ask(like(B,L))
    - or more formally, “Does Billy like Lili?”
  - Billy, like Lili! → command(like(B,L))

Sentence

- What did Gilly swallow?
  - ask(λx ∃e past(e), act(e,swallowing), swallower(e,Gilly), swallowee(e,x))
- Eat your fish!
  - command(λf act(f,eating), eater(f,Hearer), eatee(…))
- I ate my fish.
  - assert(∃e past(e), act(e,eating), eater(e,Speaker), eatee(…))

Exercise

With f(6) = 6 * 6 ,then f = λx x*x

1. With f(John) = loves(Mary, John) ,then f = ?
2. With f(John) = (∀x woman(x) ➔ loves(x, John)) then f = ?
3. With f(λx loves(Mary,x)) = (λx Obviously(loves(Mary,x))). f = ? Represent "Sue obviously loves Mary?"
4. With f(Mary)(John) = (λe act(e, loving), lovee(e, Mary), lover(e, John)). f = ?

Exercise

5. Given f as before. Assuming that

g(f(Mary)(John)) = (λe act(e, loving), lovee(e, Mary), lover(e, John), manner(e,passionate)). g = ?

Hint: write f(Mary), means “loves Mary”. g(f(Mary)) means “passionately loves Mary.“

f = λe λx λy act(e, loving), lovee(e, x), lover(e, y)

g = λf λe manner(e,passionate), f(e)

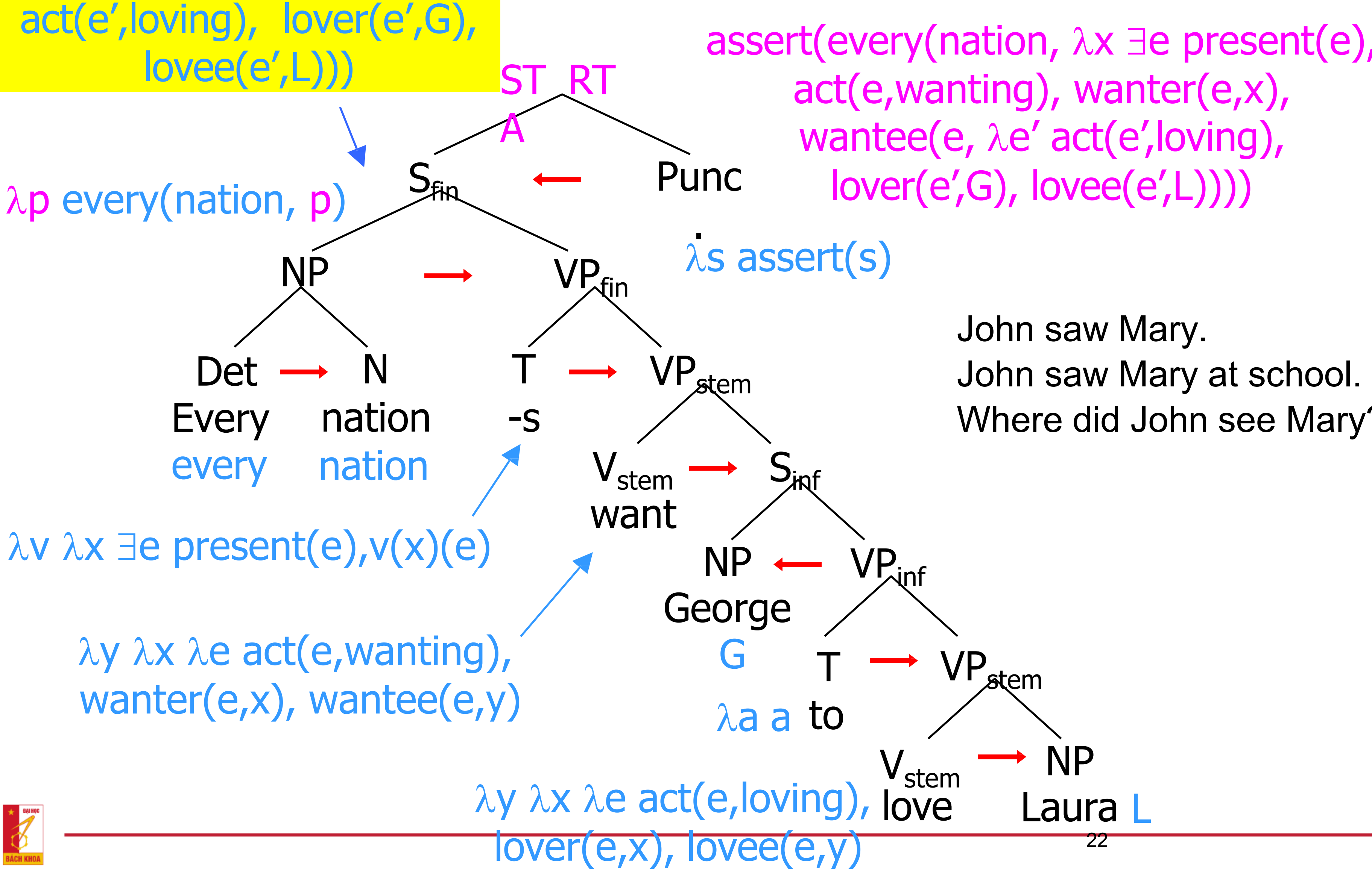
Semantic parsing

1. Syntactic parsing
2. Lexical semantic
3. Determine semantics of syntactic components, bottom-up

Chapter 18. Computational Semantics. In book Speech and Language Processing. Dan Jurafsky and James Martin. 2rd edition. Prentice Hall.

Semantic attachment

every(nation, λx ∃e present(e), act(e,wanting), wanter(e,x), wantee(e, λe′ act(e′,loving), lover(e′,G), lovee(e′,L)))



John saw Mary.

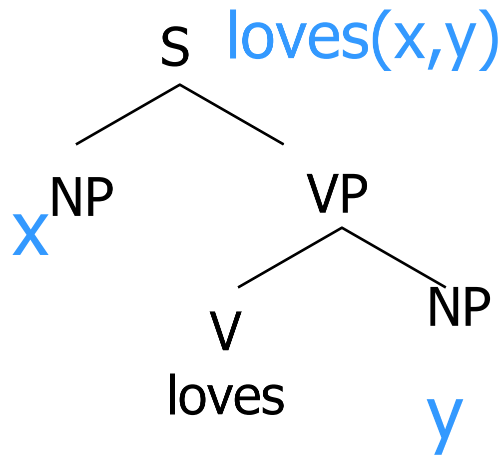
John saw Mary at school.

Where did John see Mary?

Semantic attachment

- Add “sem” attribute for each free-context rule
  - S → NP loves NP
  - S[sem=loves(x,y)] → NP[sem=x] loves NP[sem=y]
  - Semantic of S depends on semantic of NP

- TAG version:



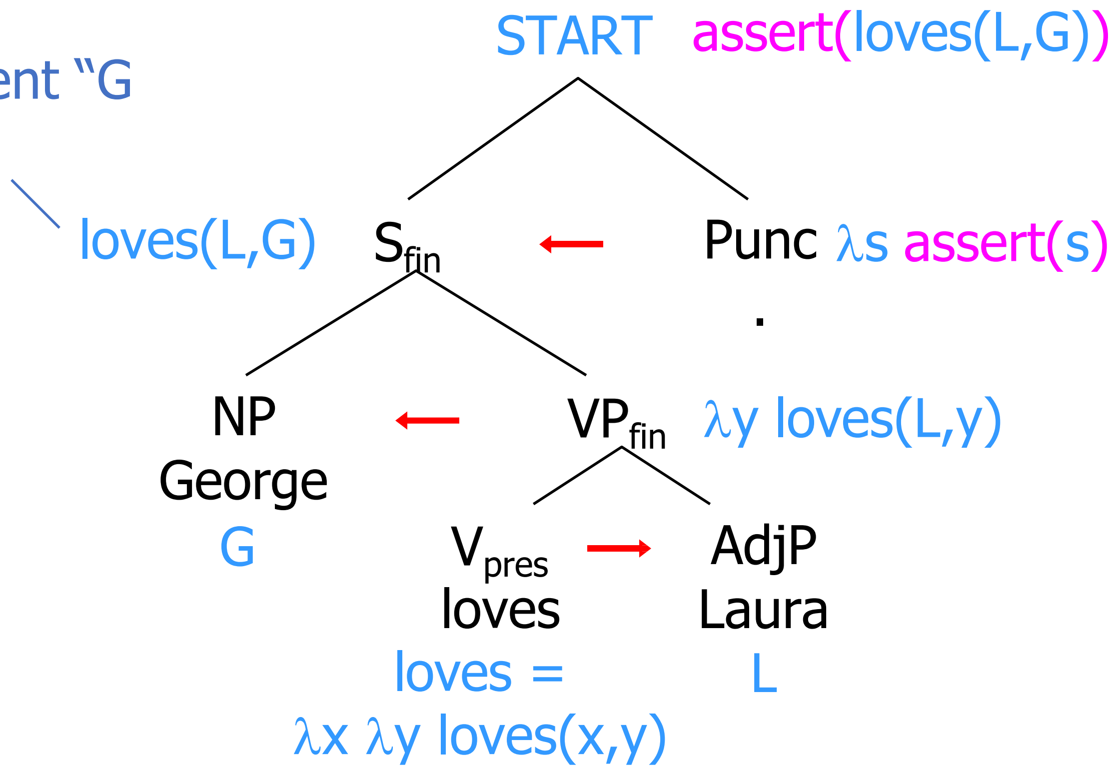
- Template filling: S[sem=showflights(x,y)] →
  I want a flight from NP[sem=x] to NP[sem=y]

Semantic attachment

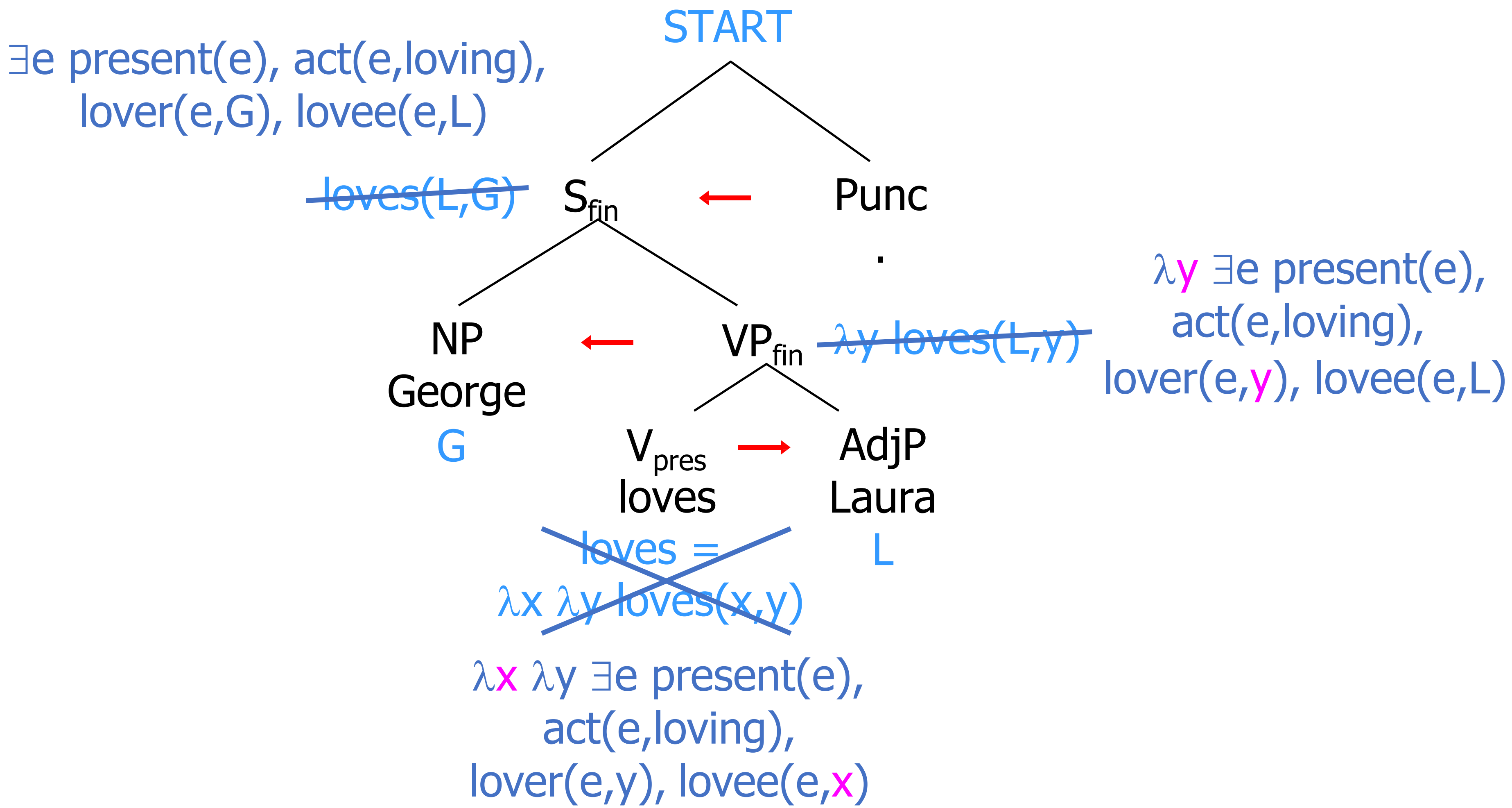
- Replace S → NP loves NP
  - S[sem=loves(x,y)] → NP[sem=x] loves NP[sem=y]
- General rule S → NP VP:
  - V[sem=loves] → loves
  - VP[sem=v(obj)] → V[sem=v] NP[sem=obj]
  - S[sem=vp(subj)] → NP[sem=subj] VP[sem=vp]
- `George loves Laura` has sem=loves(Laura)(George)
- Steps:
  - Determine semantics bottom-up
  - Chomsky-norm grammar
  - Each node has two children: 1 function and 1 param
  - To determine node semantic, apply function to param

Semantic attachment

Want to present "G loves L"



Semantic attachment



Basic meaning representation

- Use “Event”
  - (EVENT :condition1 val1 :condition2 val2… :condn valn)
- E.g:
  - (see :agent John :patient Mary :tense past)

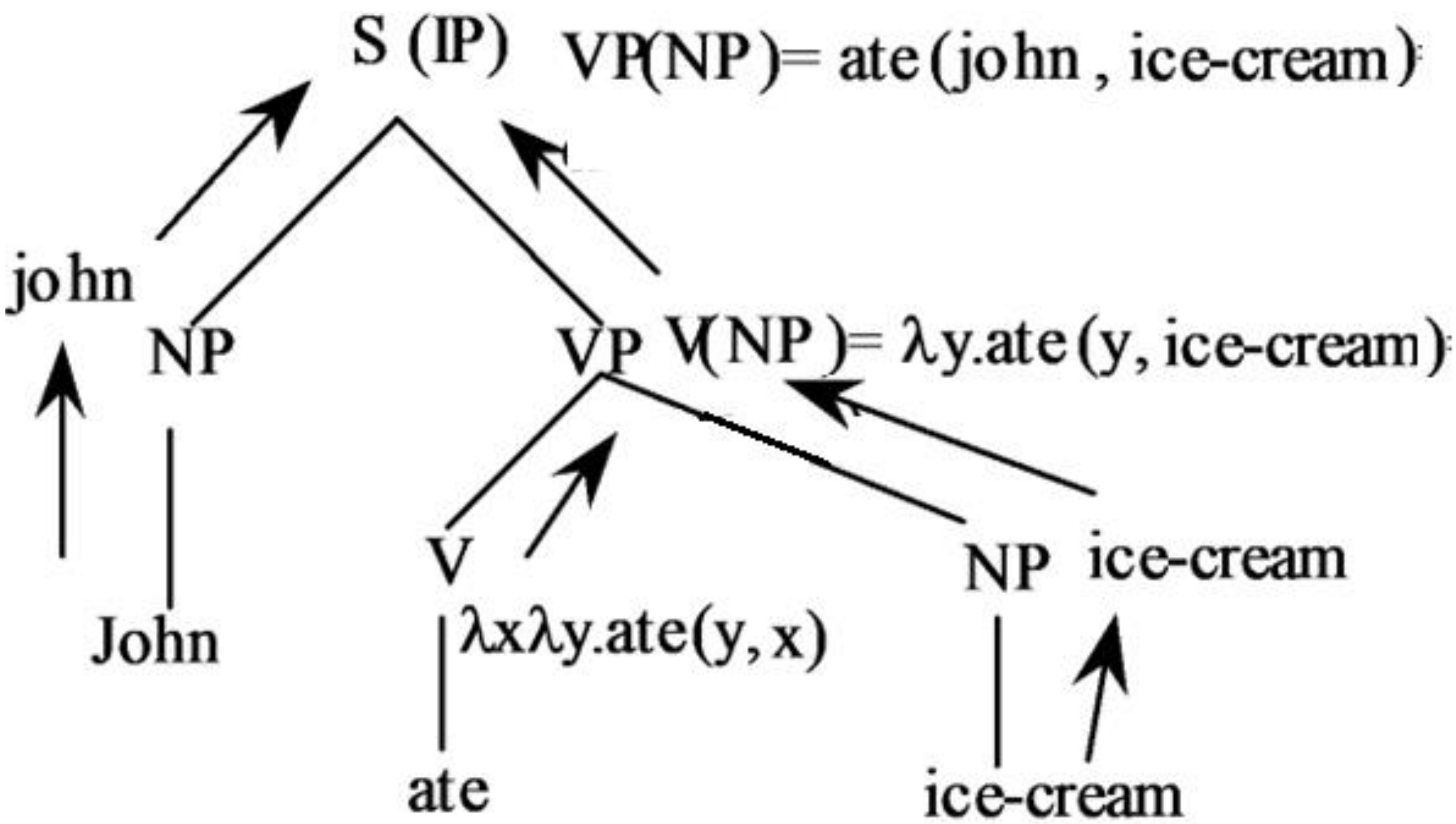
Syntax/semantic rules

Constituent/rule	Semantic
Verb *ate*	$\lambda x \lambda y.ate(y, x)$
N	N
V	V
S	S*= VP*(NP*)
NP	N*
VP	V*(NP*)

Sentence meaning

- λ form of VP is attached to λ form of NP
- Words are value
- Given syntax tree, analyze bottom-up to get sentence semantic *ate(John, ice-cream)*
- This predicate could be estimated based on DB to return a value or T/F.

Semantic translation



Example

- Provide the semantic representation of the sentences below. Explain semantic rules being used
  (in the format VP[sem=v(obj)] $\rightarrow$ V[sem=v] NP[sem=obj]).
  - Tam met An.
  - I know Tam met An.
  - Tam met An at school.

Steps

- λ at the highest level calls VP. This VP is defined at leave level using NP
- To find semantics of sentence, we call VP using NP as parameter
- At leave level, each word is accompanied by semantic information

Exercises

1. List all possible semantic representation:
   - Mai likes small cats.
   - Mai likes the cat whose name is Tom.
2. List event-based semantic representation:
   - Willy wants Lilly to get married.

Mai likes the cat whose name is Tom.

```
(ROOT
  (S
    (NP (NNP Mai))
    (VP (VBZ likes)
      (NP
        (NP (DT the) (NN cat))
        (SBAR
          (WHNP (WP$ whose) (NN name))
          (S
            (VP (VBZ is)
              (NP (NNP Tom)))))))
    (. .)))
```

Willy wants Lilly to get married.

```
(ROOT
  (S
    (NP (NNP Willy))
    (VP (VBZ wants)
      (S
        (NP (NNP Lilly))
        (VP (TO to)
          (VP (VB get)
            (ADJP (JJ married))))))
    (. .)))
```

Applications

```
(top-level)
Shall I clear the database? (y or n) y
>John saw Mary in the park.  OK.
>Where did John see Mary?
IN THE PARK.
>John gave Fido to Mary.  OK.
>Who gave John Fido?
I DON'T KNOW
>Who gave Mary Fido?  JOHN
>John saw Fido.  OK.
>Who did John see?  FIDO AND MARY
```