

---

Tecnologie del Linguaggio Naturale

Parte Prima

Lezione n. 05

Formal and Computational

Semantics

23 Marzo 2021

slides strongly based on Bill MacCartney

<http://nlp.stanford.edu/~wcmac/>

# Outline

Prendo in input l'output dell'analisi morfologica E sintattica  
In output → poter fare ragionamento automatico

- SHRDLU & Chat-80
- Computational Semantics fundamentals

- $\lambda$  abstraction

✓ composizione dei significati / computational semantics

- CS per:

—> Articoli indeterminativi —> Nomi propri —> Verbi transitivi

—> Cordinazione dei nomi —> Quantificatori universali

- NLTK ← libreria di Python

- Altri approcci alla semantica

- AMR

- Filler-slot

- Semantic Grammars

# Outline

---

- SHRDLU & Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:

->Articoli indeterminativi ->Nomi propri ->Verbi transitivi

->Cordinazione dei nomi ->Quantificatori universali

- NLTK
- Altri approcci alla semantica

# Si possono costruire le grammatiche a mano?

un po' di storia

- Per l'italiano: L. Lesmo (1985 -> 2014), Common Lisp
- Per l'inglese:
  - SHRDLU (Winograd 1972)
    - <https://www.youtube.com/watch?v=8SvD-INg0TA>
    - <http://hci.stanford.edu/winograd/shrdlu/>
  - CHAT-80: 1979-82 F. Pereira & D. Warren, Prolog
    - Linguaggio naturale per un DB sulla geografia
      - Where is Rome?
      - Which country bordering the Mediterranean borders a country that is bordered by a country whose population exceeds the population of India? <http://nli-gems.org/system.php?id=chat-80>

esempio



# Prolog :)

```
/* Sentences */
sentence(S) --> declarative(S), terminator(.) .
sentence(S) --> wh_question(S), terminator(?) .
sentence(S) --> yn_question(S), terminator(?) .
sentence(S) --> imperative(S), terminator(!) .

/* Noun Phrase */
np(np(Agmt,Pronoun,[]),Agmt,NPCase,def,_,Set,Nil) -->
{is_pp(Set)},
pers_pron(Pronoun,Agmt,Case),
{empty(Nil), role(Case,decl,NPCase)}.

/* Prepositional Phrase */
pp(pp(Prep,Arg),Case,Set,Mask) -->
prep(Prep),
{prep_case(NPCase)},
np(Arg,_,NPCase,_,Case,Set,Mask).
```

# Outline

---

- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:
  - Articoli indeterminativi
  - Nomi propri
  - Verbi transitivi
  - Corrdinazione dei nomi
  - Quantificatori universali
- NLTK

dietro ad Prolog c'è l'idea fondamentale di costruire il significato dell'insieme a partire dal significato delle parti

# Rappresentare il significato

- Quale forma per il significato?

- DB tables, SQL, description logics, modal logics, AMR, frames, ...

*La forma è semplice e intelligenza degli esseri (owl)*

- Blackburn & Bos -> first-order logic (FoL)

- Paolo corre -> run(Paolo)

*logica dei predicati  
è un linguaggio dichinativo  
“da una fotografia del mondo,  
ma non dà l’ordine con cui deve farsi”*

- Paolo ama Francesca -> love(Paolo, Francesca)

- Tutti gli uomini amano Francesca ->  $\forall x (\text{man}(x) \rightarrow \text{love}(x, \text{Francesca}))$

- Il problema nasce per le “frasi incomplete”

- ama -> love(x,y) *sono variabili libere, non ci sono quantificatori* → non c'è un formula ben fondata

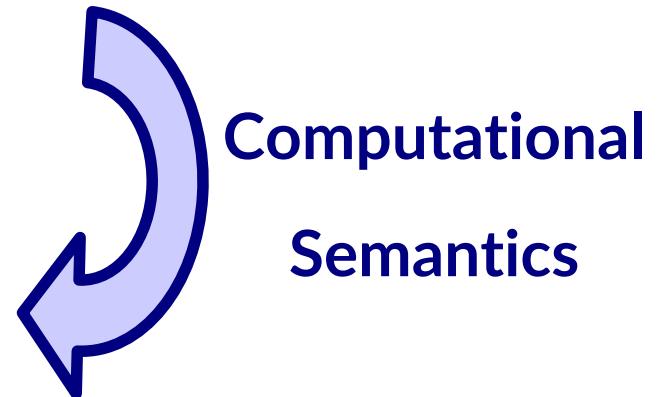
- Lambda-FoL termini per parole e sintagmi -> FoL per le frasi

*è un linguaggio e una rappresentazione statica, non permette di fare algoritmi  
e conoscere gli oggetti del mondo  
vs Lambda calcolo: è un calcolo, ha una dinamica → a partire da ogg. elementi permette di costruire oggetti + complessi*

# Computational Semantics

→ *lo scopo è  
fare ragionare  
automaticamente*

- Frase
  - Paolo corre. Tutti quelli che corrono vivono meglio.
- Syntactic analysis
  - $(S (NP \text{ Paolo}) (VP \text{ corre}))$
- Semantic interpretation/analysis
  - $\text{run(paolo)}$
- Inference
  - $\forall x.\text{run}(x) \rightarrow \text{goodLife}(x)$
  - $\text{run(paolo)} \rightarrow \text{goodLife(paolo)}$



# FoL syntax in una slide

- FOL simboli

- Costanti: **john, mary**
- Predicati & relazioni: **man, walks, loves**
- Variabili: **x, y**
- Connettivi Logici:  $\wedge \vee \neg \rightarrow$
- Quantificatori:  $\forall \exists$
- Altri simboli ausiliari: **( ) ,**

substitui a ogg. del mondo che  
voglio modellare



## Content Words



## Function Words

- FOL formule

- Formule atomiche: loves(paolo,francesca)
- Formule composte: man(paolo)  $\wedge$  loves(paolo,francesca)
- Formule quantificate:  $\exists x$  (man(x))

serà p: vero o falso

Perche' non si usa la Second Order logic? Completaz. Troppo complesse  
la FoL è un buon compromesso tra espressività e complessità computaz.

# Il mondo dei ristoranti

modello ↴

## Domain

- razi → Matthew, Franco, Katie and Caroline  
ristoranti → Frasca, Med, Rio  
tipologie → Italian, Mexican, Eclectic

$$\mathcal{D} = \{a, b, c, d, e, f, g, h, i, j\}$$
$$a, b, c, d$$
$$e, f, g$$
$$h, i, j$$

## Properties

- Noisy ↗ *Predicato*  
Frasca, Med, and Rio are noisy

$$\text{Noisy} = \{e, f, g\}$$
 ← "Cosa è Noisy"

## Relations

- Likes  
Matthew likes the Med  
Katie likes the Med and Rio  
Franco likes Frasca  
Caroline likes the Med and Rio

$$\text{Likes} = \{\langle a, f \rangle, \langle c, f \rangle, \langle c, g \rangle, \langle b, e \rangle, \langle d, f \rangle, \langle d, g \rangle\}$$

## Serves

- Med serves eclectic  
Rio serves Mexican  
Frasca serves Italian

$$\text{Serves} = \{\langle e, j \rangle, \langle f, i \rangle, \langle e, h \rangle\}$$

interpretazione = costruire degli informi

# FoL e ristoranti

<i>Formula</i>	$\rightarrow$	<i>AtomicFormula</i>
		<i>Formula Connective Formula</i>
		<i>Quantifier Variable, ... Formula</i>
		$\neg$ <i>Formula</i>
		( <i>Formula</i> )
<i>AtomicFormula</i>	$\rightarrow$	<i>Predicate(Term, ...)</i>
<i>Term</i>	$\rightarrow$	<i>Function(Term, ...)</i>
		<i>Constant</i>
		<i>Variable</i>
<i>Connective</i>	$\rightarrow$	$\wedge$   $\vee$   $\Rightarrow$
<i>Quantifier</i>	$\rightarrow$	$\forall$   $\exists$
<i>Constant</i>	$\rightarrow$	<i>A</i>   <i>VegetarianFood</i>   <i>Maharani</i> ...
<i>Variable</i>	$\rightarrow$	<i>x</i>   <i>y</i>   ...
<i>Predicate</i>	$\rightarrow$	<i>Serves</i>   <i>Near</i>   ...
<i>Function</i>	$\rightarrow$	<i>LocationOf</i>   <i>CuisineOf</i>   ...

↑  
sintassi della FoL

# Un modello per i ristoranti è ...

---

Abbiamo bisogno di mappare gli elementi di significato  
(formule FoL) al mondo

- FoL Termini -> elementi del dominio
  - Med ->  $f$
- FoL formule atomiche -> insiemi sui domini di elementi o insiemi di tuple
- $\text{noisy}(\text{Med})$  è vera se  $f$  è nell'insieme degli elementi della relazione **noisy**
- $\text{near}(\text{Med}, \text{Rio})$  è vera se la tupla  $\langle f, g \rangle$  è nell'insieme delle tuple che corrispondono a “**near**” nell'interpretazione

# Un modello per i ristoranti è ...

---

- Per formule più grandi (1. connettivi) usiamo le tavole di verità

$P$	$Q$	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>	<i>False</i>	<i>True</i>
<i>False</i>	<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>
<i>True</i>	<i>False</i>	<i>False</i>	<i>False</i>	<i>True</i>	<i>False</i>
<i>True</i>	<i>True</i>	<i>False</i>	<i>True</i>	<i>True</i>	<i>True</i>

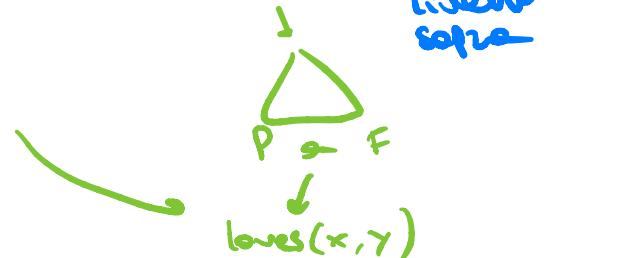
- Per formule più grandi (2. quantificatori) usiamo la semantica di *per ogni* e di *esiste* ...

voglio arrivare a comporre la semantica della frase (=della radice)

# CS Fundamental Algorithm

↑ "computational semantics"

- ① Parsificare la frase per ottenere l'abero sintattico (costituenti) ← viene dal livello sopra
- ② Cercare la semantica di ogni parola nel lessico
- ③ Costruire la semantica per ogni sintagma
  - Bottom-up
  - Syntax-driven: "rule-to-rule translation"



sintattico

## Principio di composizionalità di Frege

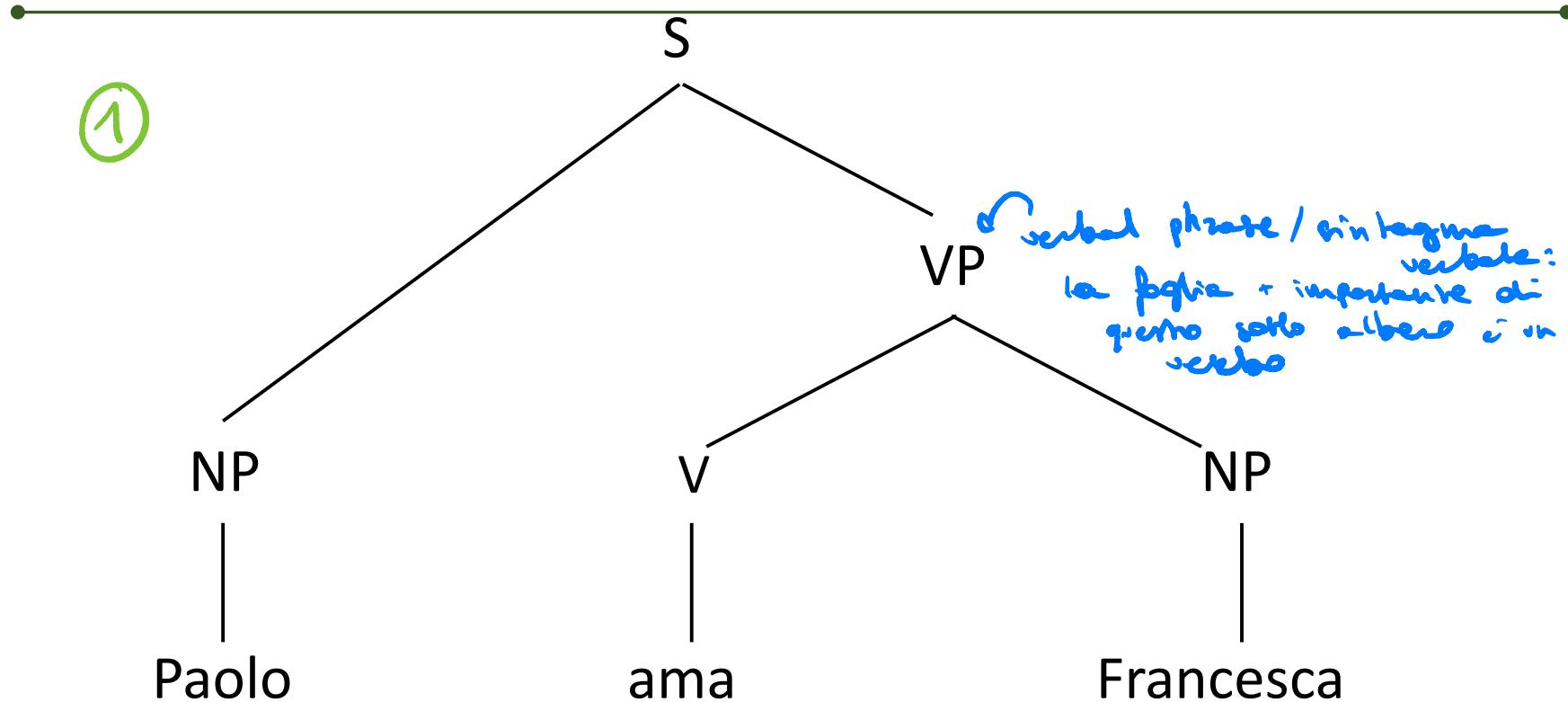
Il significato del tutto è determinato dal significato delle parti e dalla maniera in cui sono combinate

la semantica + complesse è quella delle function words, + che delle content words

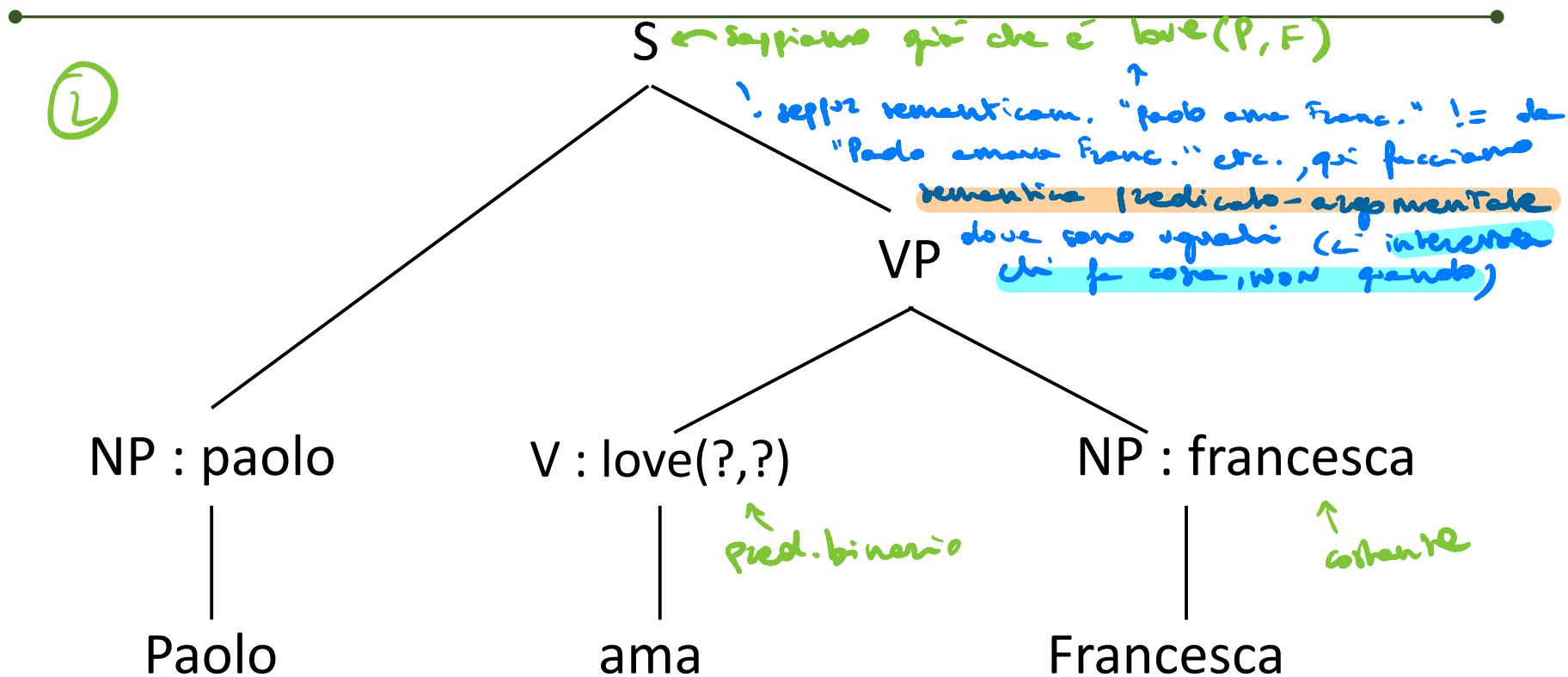
"bottiglia di vino" → "bottiglia vino"

← c'è una sorta di trasformazione

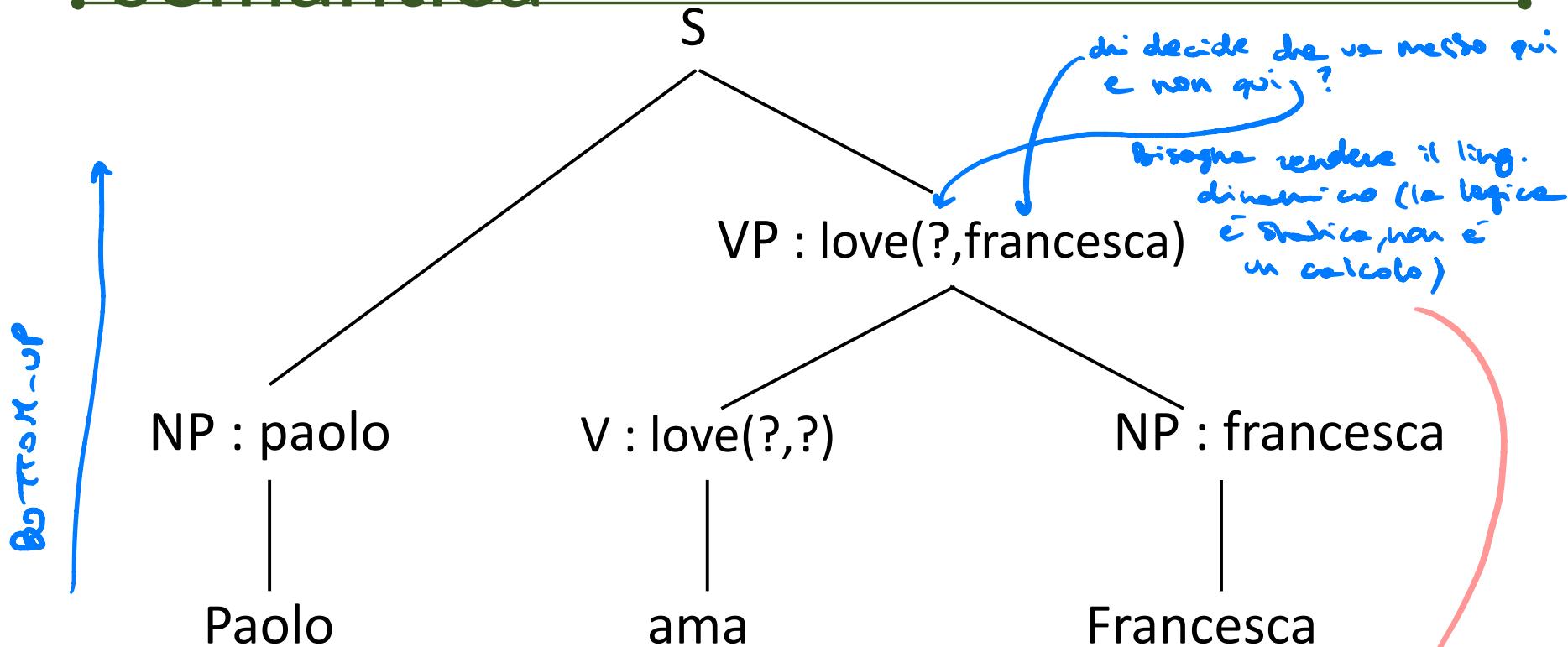
# Esempio: analisi sintattica



# Esempio: lessico semantico



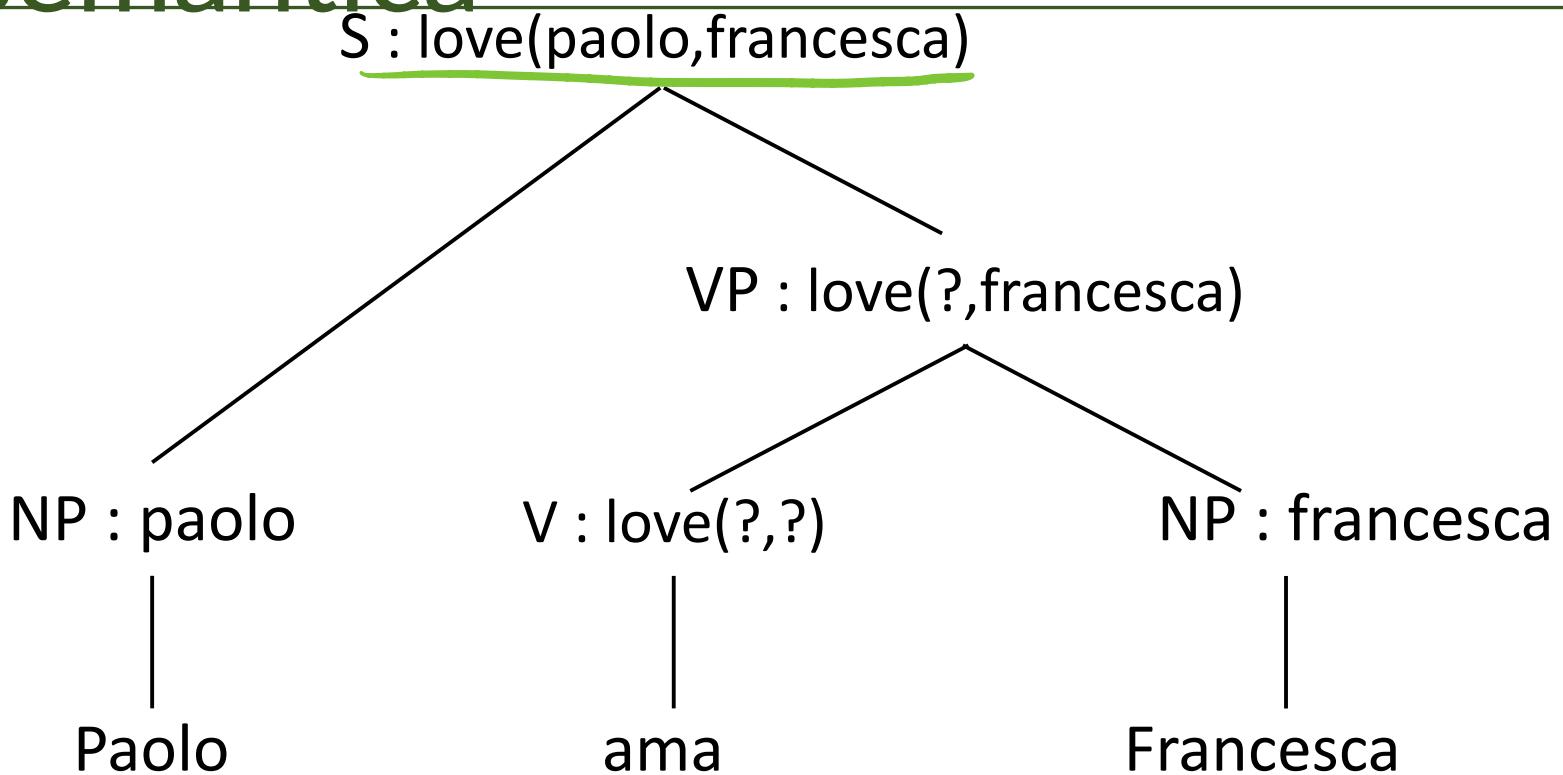
# Esempio: composizione semantica



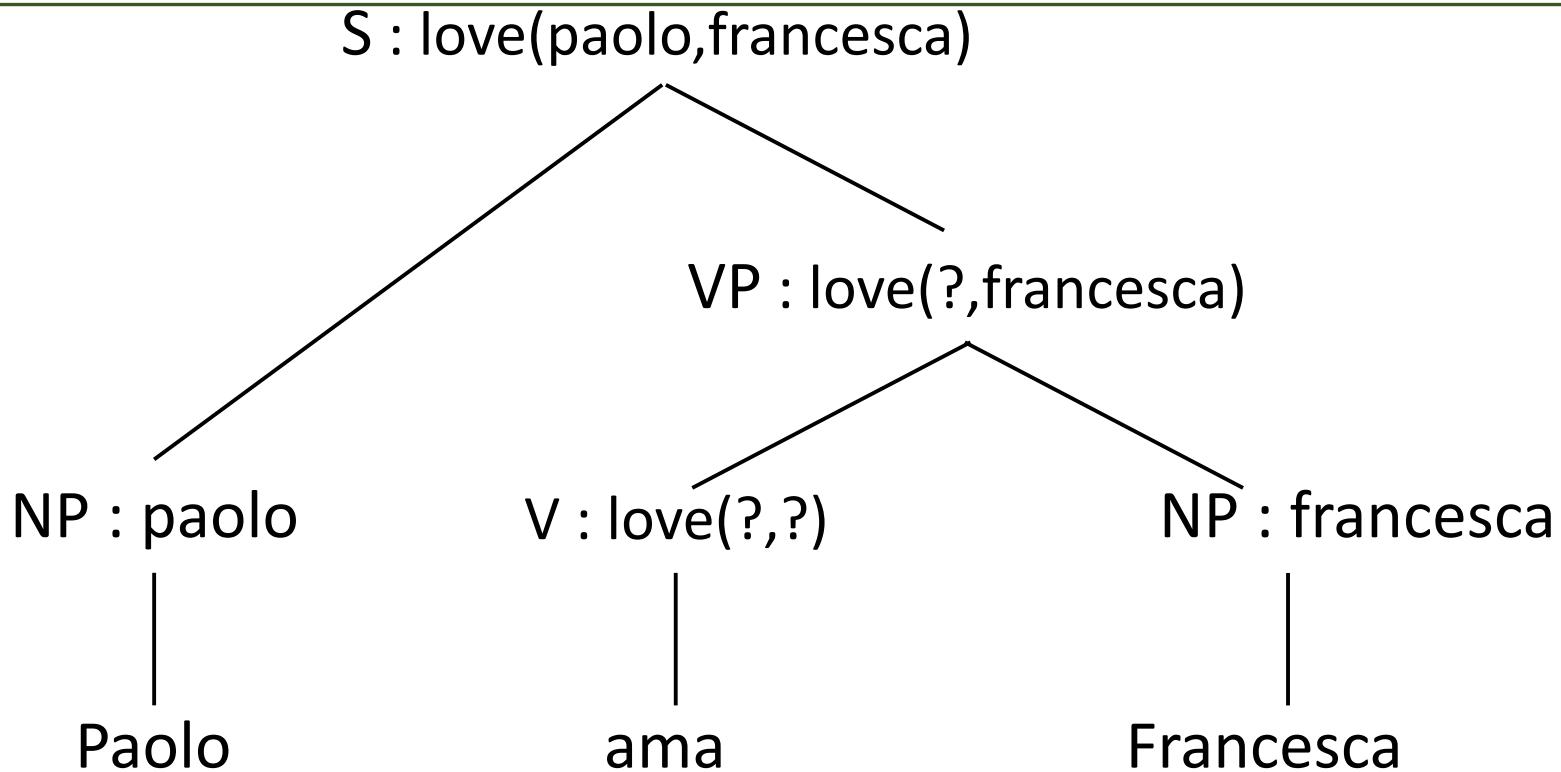
Quali sono i problemi della FOL che ci hanno fatto introdurre i quantificatori?

- non so come scrivere  $\text{love}(?,?)$ , perché in FOL non so che quantificatori usare
- come stabilire l'ordine con cui vanno riempiti gli argomenti dei predicati?

# Esempio: composizione semantica



# Composizionalità

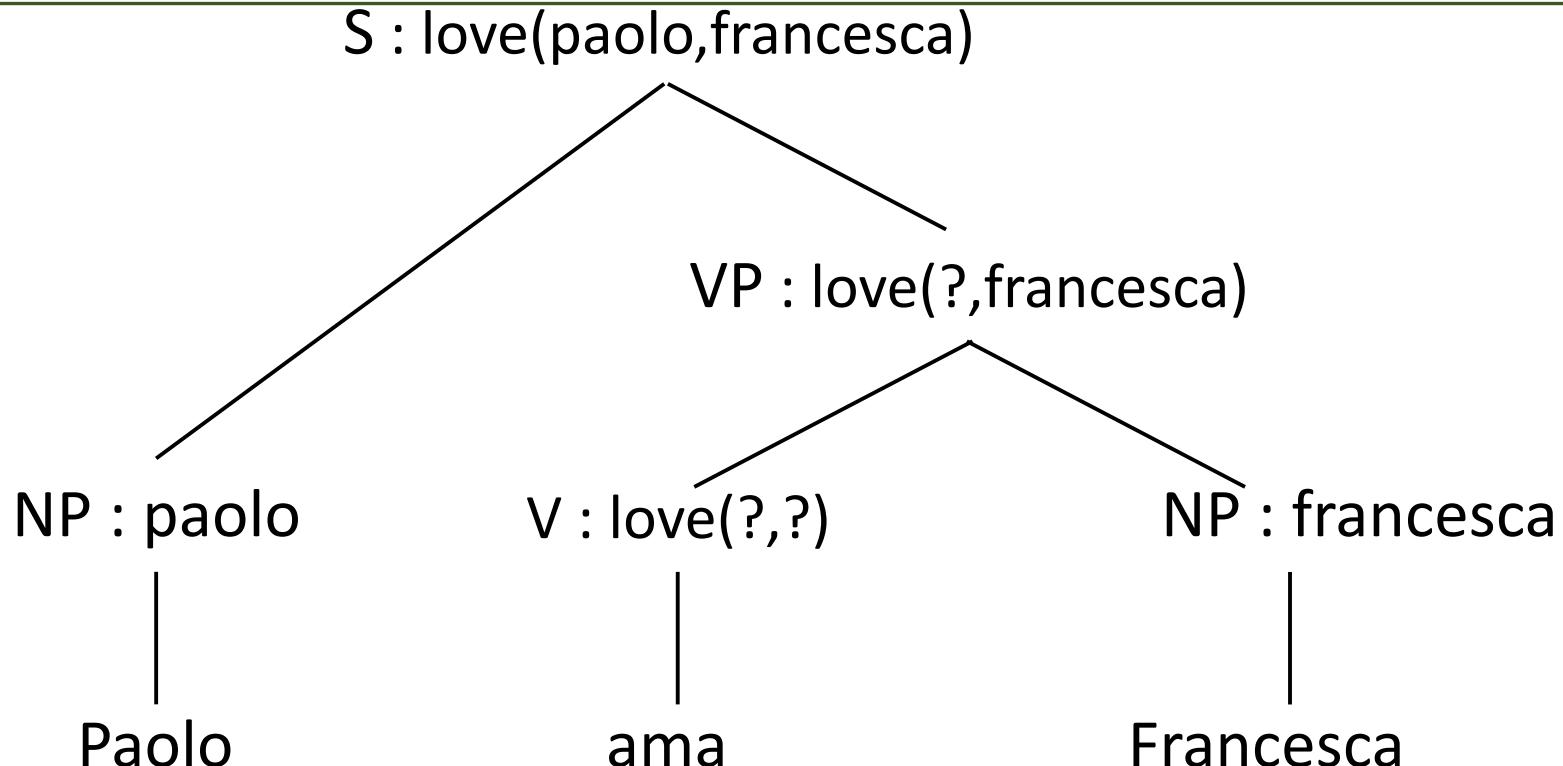


Il significato della frase è costruito:

- dal significato delle parole -> lessico
- risalendo le costruzioni sintattiche -> regole semantiche

# Sistematicità

l'algoritmo deve funzionare sempre,  
x tutte le frasi

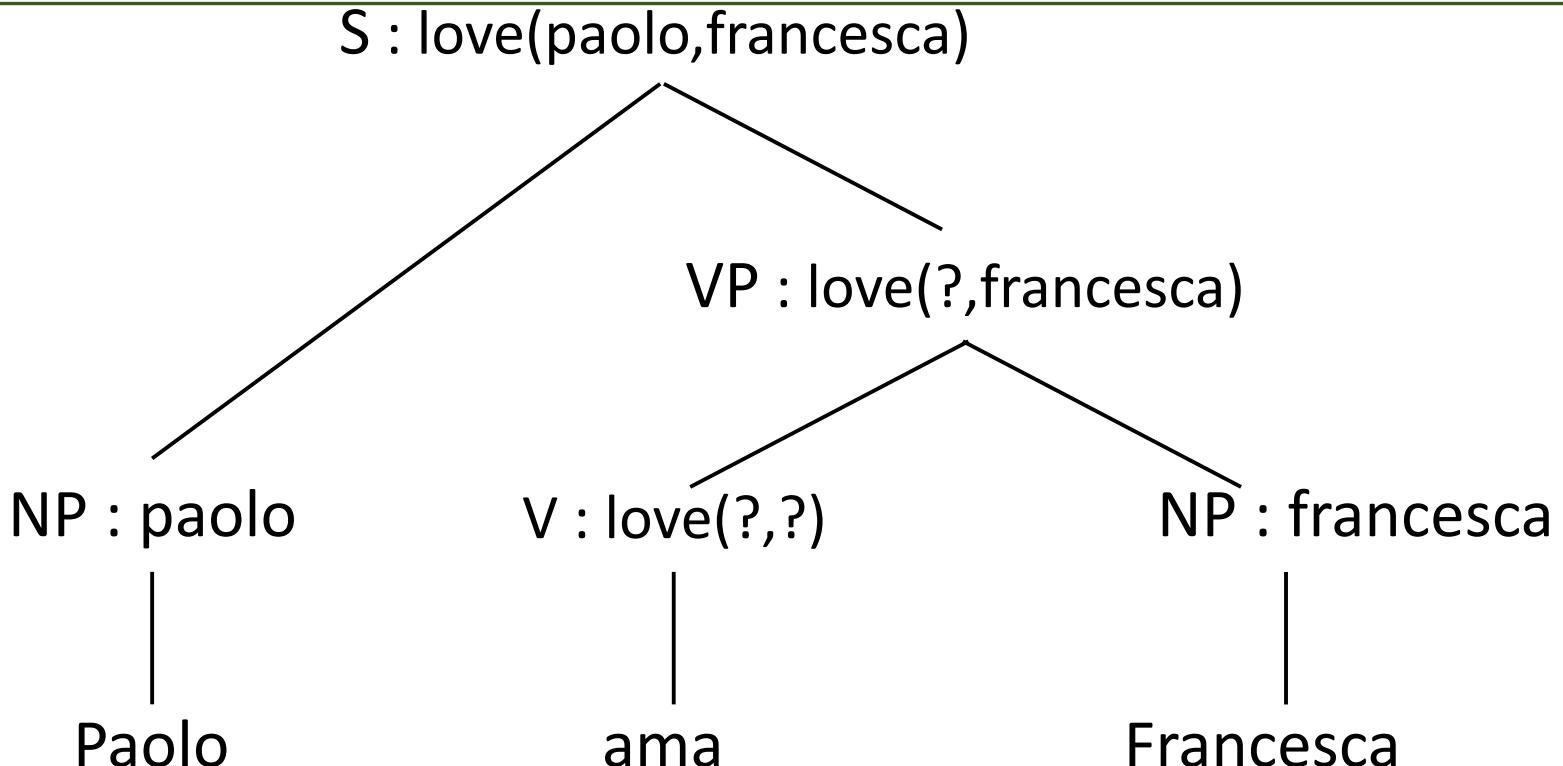


1. Come costruiamo il significato del VP?

$\text{love}(?,\text{francesca})$  o  $\text{love}(\text{francesca},?)$

2. Come specificare in che maniera combinare i pezzi?

# Sistematicità



3. Come rappresentare pezzi di formule?

  $\text{love}(?,\text{francesca}) \rightarrow \text{non FoL}$ ,  $\text{love}(x,\text{francesca}) \rightarrow \text{variabile libera?}$

4. Vorremmo: *Tutti amano Francesca*  $\rightarrow \forall x (\text{love}(x,\text{francesca}))$

# Outline

---

- Chat-80
- Computational Semantics fundamentals
  - λ abstraction *: il calcolo trasforma gli oggetti (love(!,?)) in delle funzioni*
  - CS per:
    - >Articoli indeterminativi ->Nomi propri ->Verbi transitivi
    - >Cordinazione dei nomi ->Quantificatori universali
- NLTK
- Altri approcci alla semantica

# Lambda abstraction

*funzione anonima, che può essere usata come argomento  
di un'altra funzione*

- Nuovo operatore  $\lambda$  per legare (bind) le variabili libere
  - $\lambda x.\text{love}(x, \text{francesca}) \rightarrow \text{amare Francesca}$  *diventa una funzione!*
- Il nuovo simbolo meta-logico  $\lambda$  segna l'informazione mancante nella meta-formula  $\lambda$ -FoL.  $\rightarrow$  **astrarre su  $x$**
- Programmazione:
  - Common Lisp: `((lambda (x) (+ 1 x)) 7)`
    - [ McCarthy58–Russel60 ]
  - Python: `lambda x: x % 2 == 0`
  - Java 8:

<http://www.oracle.com/webfolder/technetwork/tutorials/obe/java/Lambda-QuickStart/index.html>

- Come combinare  $\lambda$ -FoL e FoL?

# Function application

- $(\lambda x.\text{love}(x, \text{mary})) @ \text{john} \Rightarrow \text{love}(\text{john}, \text{mary})$

||  
||

- $(\lambda x.\text{love}(x, \text{mary}))(john)$

- La FA attiva un'operazione elementare chiamata

- Beta reduction -> ...



# Beta reduction

sono di operaz. elementari che prendono le funz. e permettono effettivamente di fare la function application

funzione      arg.  
 $(\lambda x.\text{love}(x, \text{mary})) (\text{john})$

- ① Elimina il  $\lambda$

$(\text{love}(x, \text{mary})) (\text{john})$

- ② Rimuovi l'argomento

$\text{love}(x, \text{mary})$

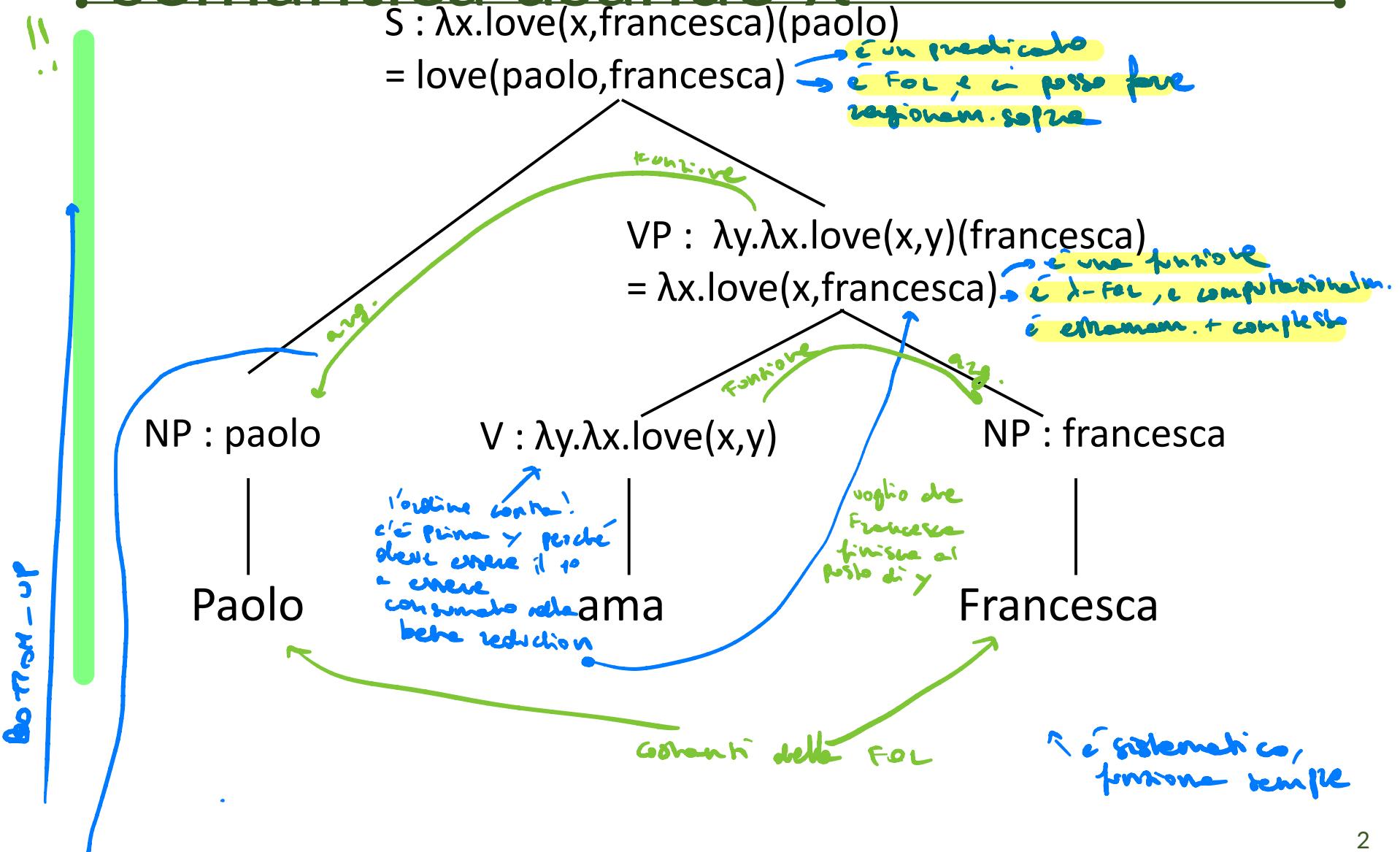
- ③ Rimpiazza le occorenze della variabile legata dal  $\lambda$  con l'argomento in tutta la formula

$\text{love}(\text{john}, \text{mary})$



# Esempio: composizione semantica usando $\lambda$

le chiede spesso  
↓



# Una semplicissima grammatica semantica (1)

*o meglio, c'è una gramm. sintattica in cui...  
aggiungo un attributo, che contiene la semantica*

## Lessico

NP : paolo  $\rightarrow$  Paolo

NP : francesca  $\rightarrow$  Francesca

V :  $\lambda y. \lambda x. \text{love}(x, y)$   $\rightarrow$  ama

*regole delle context-free*

*← le foglie*

## Regole di composizione

VP : f(a)  $\rightarrow$  V : f      NP : a

*← x : nodi intermedi*

S : f(a)  $\rightarrow$  NP : a VP : f

# Una semplicissima grammatica semantica (1)

## Lessico

NP : paolo -> Paolo

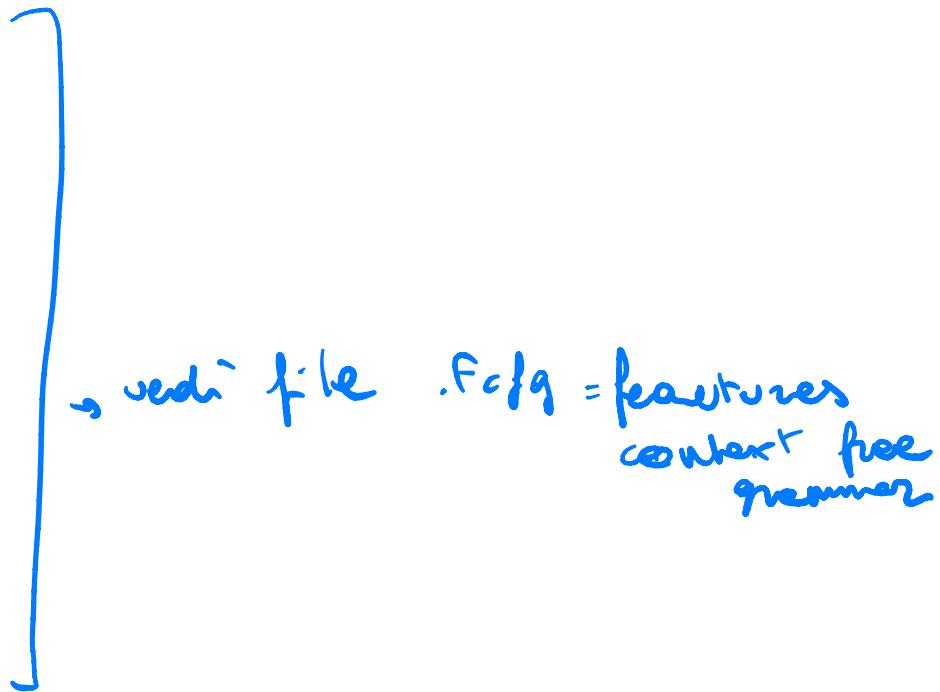
NP : rancesca -> Francesca

V :  $\lambda y.\lambda x.\text{love}(x, y)$  -> ama

## Regole di composizione

VP : f(a) -> V : f    NP : a

S : f(a) -> NP : a VP : f



Augmented CFG == Grammatiche ad attributi

YACC expr:expr '\*' expr { \$\$=\$1\*\$3};

# Semantica di Montague

---

Richard Montague (1930-71)

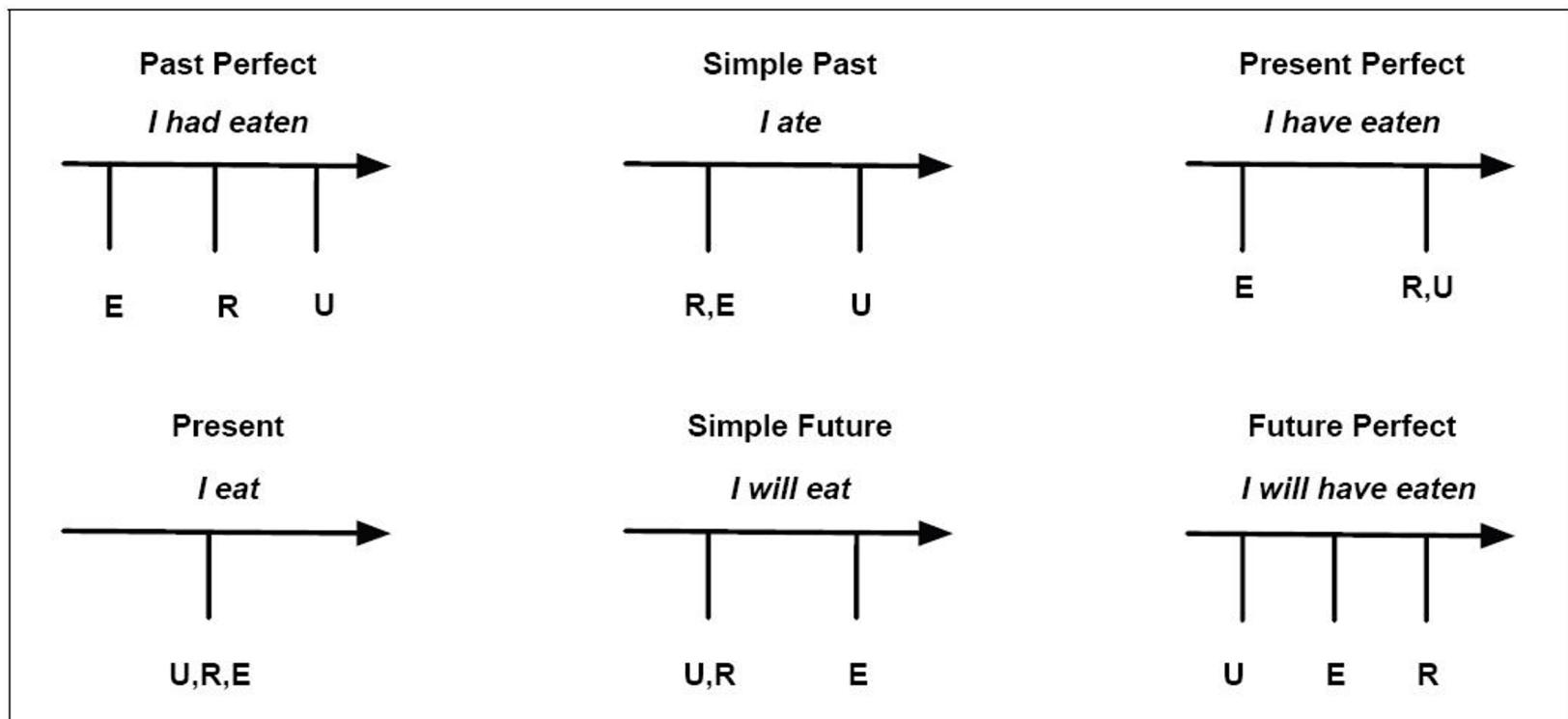


*... I reject the contention that an important theoretical difference exists between formal and natural languages ...*

English as formal language, pp. 188-221, Linguaggi nella Società e nella Tecnica. Convegno promosso dalla Ing. C. Olivetti & C. S. p. A. per il centenario della nascita di Camillo Olivetti, Museo Nazionale della Scienza e della Tecnica, Milano, 14-17 Ottobre 1968 (Ristampato, Edizioni di Comunità, 1970)

# Stiamo considerando solo parte del significato!

- ❖ Struttura Predicato-Argomento
- ❖ Il tempo? -> Reference time: Reichenbach, 1947



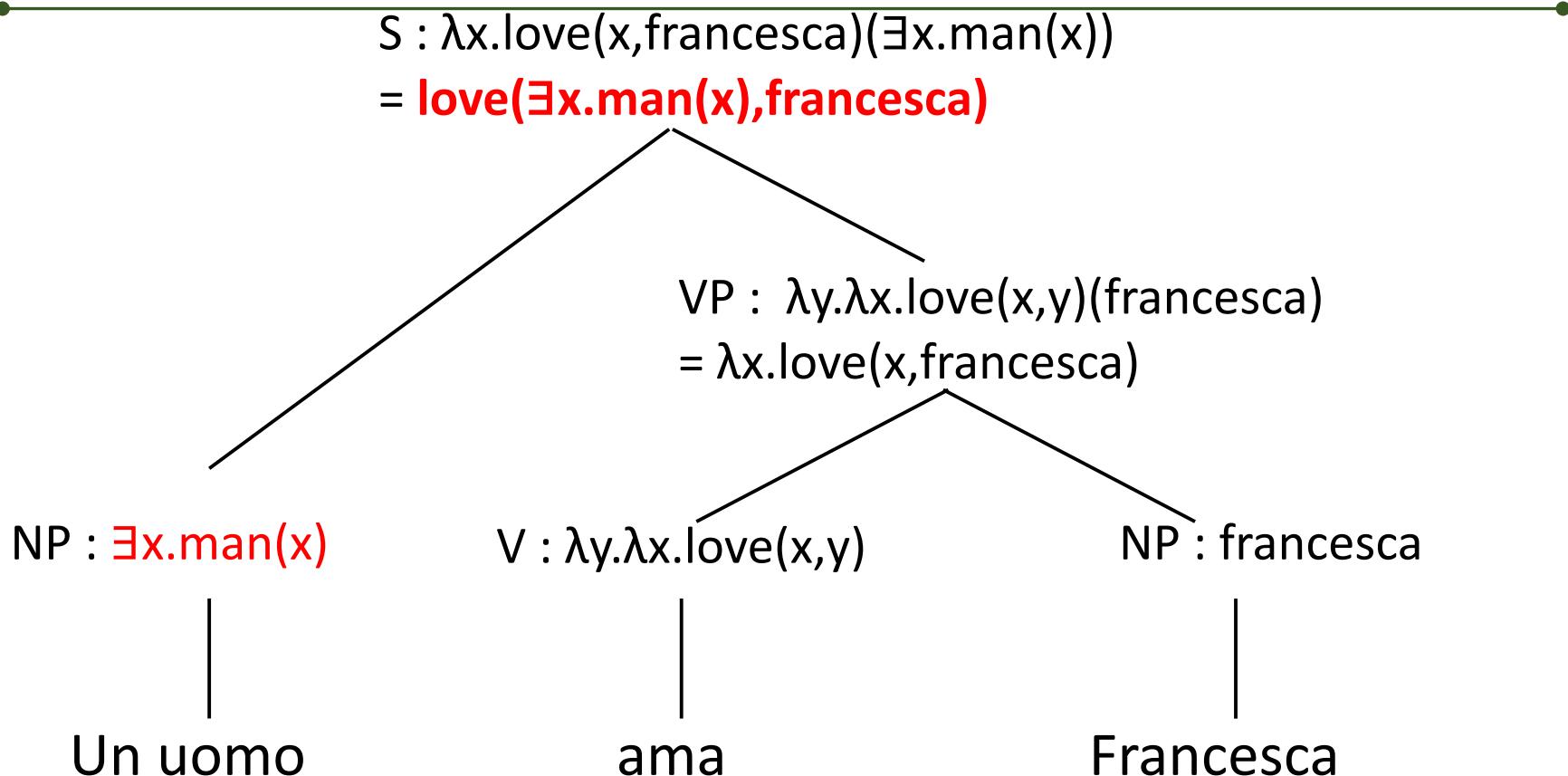
# Outline

---

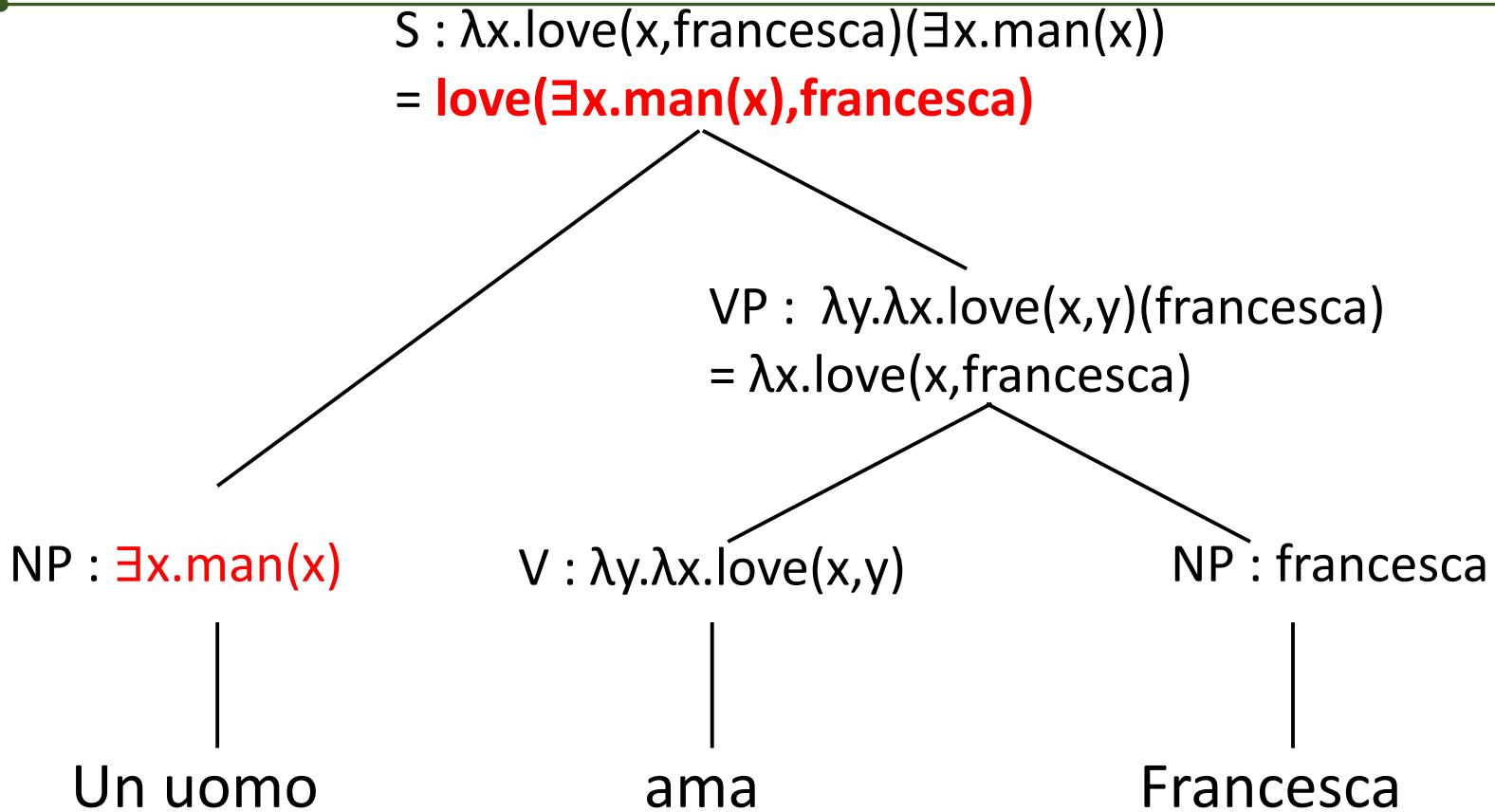
- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:
  - >Articoli indeterminativi->Nomi propri->Verbi transitivi
  - >Cordinazione dei nomi->Quantificatori universali
- NLTK
- Altri approcci alla semantica



## Esempio 2: gli articoli



# Esempio 2: gli articoli



$\exists x.\text{man}(x)$  significa “c'è un uomo ...” e non “un uomo”

# Come trattare gli articoli -> .reverse enginnering

---

*L'uomo saggio inizia dalla fine e finisce col principio*  
(cit. ?)

Vorremmo:

Un uomo ama Francesca =>

$\exists z (\text{man}(z) \wedge \text{love}(z, \text{francesca}))$

$\exists z (\lambda y. \text{man}(y)(z) \wedge \lambda x. \text{love}(x, \text{francesca})(z))$

Dobbiamo permettere l'astrazione anche sui predicati:

$(\lambda Q. \exists z (\lambda y. \text{man}(y)(z) \wedge Q(z))) (\lambda x. \text{love}(x, \text{francesca}))$

$(\lambda P. \lambda Q. \exists z (P(z) \wedge Q(z))) (\lambda x. \text{love}(x, \text{francesca})) (\lambda y. \text{man}(y))$

# Come trattare gli articoli

---

Nel lessico:

$$DT : \lambda P. \lambda Q. \exists z (P(z) \wedge Q(z)) \rightarrow un$$

# Come trattare gli articoli

---

Nel lessico:

DT :  $\lambda P. \lambda Q. \exists z (P(z) \wedge Q(z)) \rightarrow \text{un}$

DT :  $\lambda P. \lambda Q. \forall z (P(z) \rightarrow Q(z)) \rightarrow \text{tutti}$

DT :  $\lambda P. \lambda Q. \forall z (P(z) \rightarrow \neg Q(z)) \rightarrow \text{nessun}$

# Una semplicissima grammatica semantica (2)

---

## Lessico

DT :  $\lambda P. \lambda Q. \exists z (P(z) \wedge Q(z)) \rightarrow \text{un}$

N :  $\lambda y. \text{man}(y) \rightarrow \text{uomo}$

NP : francesca  $\rightarrow$  Francesca

V :  $\lambda y. \lambda x. \text{love}(x, y) \rightarrow \text{ama}$

## Regole di composizione

VP : f(a)  $\rightarrow$  V : f     NP : a

S : f(a)  $\rightarrow$  NP : f   VP : a

# Esempio 2: gli articoli

$$\begin{aligned} S &: (\lambda Q. \exists z (\text{man}(z) \wedge Q(z))) (\lambda x. \text{loves}(x, \text{francesca})) \\ &= \exists z (\text{man}(z) \wedge (\lambda x. \text{loves}(x, \text{francesca}))(z)) \\ &= \exists z (\text{man}(z) \wedge \text{loves}(z, \text{francesca})) \end{aligned}$$

$$\begin{aligned} \text{NP} &: (\lambda P. \lambda Q. \exists z (P(z) \wedge Q(z))) (\lambda y. \text{man}(y)) \\ &= \lambda Q. \exists z ((\lambda y. \text{man}(y))(z) \wedge Q(z)) \\ &= \lambda Q. \exists z (\text{man}(z) \wedge Q(z)) \end{aligned}$$

$$\begin{aligned} \text{VP} &: \lambda y. \lambda x. \text{love}(x, y)(\text{francesca}) \\ &= \lambda x. \text{love}(x, \text{francesca}) \end{aligned}$$

$$\text{DT}: \lambda P. \lambda Q. \exists z (P(z) \wedge Q(z))$$

$$N : \lambda y. \text{man}(y)$$

$$V : \lambda y. \lambda x. \text{love}(x, y)$$

$$NP : \text{francesca}$$

Un

uomo

ama

Francesca

# Esempio 2: gli articoli

$$\begin{aligned} S &: (\lambda Q. \exists z (man(z) \wedge Q(z))) (\lambda x. loves(x, francesca)) \\ &= \exists z (man(z) \wedge (\lambda x. loves(x, francesca))(z)) \\ &= \exists z (\text{man}(z) \wedge \text{loves}(z, francesca)) \end{aligned}$$

$$\begin{aligned} NP &: (\lambda P. \lambda Q. \exists z (P(z) \wedge Q(z))) (\lambda y. man(y)) \\ &= \lambda Q. \exists z ((\lambda y. man(y))(z) \wedge Q(z)) \\ &= \lambda Q. \exists z (man(z) \wedge Q(z)) \end{aligned}$$

$$\begin{aligned} VP &: \lambda y. \lambda x. love(x, y)(francesca) \\ &= \lambda x. love(x, francesca) \end{aligned}$$

$$DT: \lambda P. \lambda Q. \exists z (P(z) \wedge Q(z))$$

$$N: \lambda y. man(y)$$

$$V: \lambda y. \lambda x. love(x, y)$$

$$NP: francesca$$

Un

uomo

ama

Francesca

“Paolo ama Francesca” funziona **ancora?**

$\lambda x. love(x, francesca) @ (paolo)$  **VS.**  $(paolo) @ (\lambda x. love(x, francesca))$

# Outline

---

- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:

->Articoli indeterminativi ->**Nomi propri** ->Verbi transitivi

->Cordinazione dei nomi ->Quantificatori universali

- NLTK
- Altri approcci alla semantica

# Problema con i nomi propri .soggetto

---

S : f(a) -> NP : f VP : a

(paolo) @ ( $\lambda x.\text{love}(x, \text{francesca})$ )

# Problema con i nomi propri .soggetto

~~S : f(a) → NP : f VP : a~~

~~(paolo) @ ( $\lambda x.\text{love}(x, \text{francesca})$ )~~

$\lambda P.P(\text{paolo})$

$\lambda P.P(\text{paolo}) @ (\lambda x.\text{love}(x, \text{francesca})) \rightarrow$

$\lambda x.\text{love}(x, \text{francesca})) @ (\text{paolo}) \rightarrow$

$\text{love}(\text{paolo}, \text{francesca}))$

Type-raising

old type: e

new type:  $(e \rightarrow t) \rightarrow t$

# Una semplicissima grammatica semantica (3)

---

## Lessico

DT :  $\lambda P. \lambda Q. \exists z (P(z) \wedge Q(z)) \rightarrow \text{un}$

N :  $\lambda y. \text{man}(y) \rightarrow \text{uomo}$

NP :  $\lambda P. P(\text{paolo}) \rightarrow \text{Paolo}$

NP :  $\lambda P. P(\text{francesca}) \rightarrow \text{Francesca}$

V :  $\lambda y. \lambda x. \text{love}(x, y) \rightarrow \text{ama}$

## Regole di composizione

VP :  $f(a) \rightarrow V : f \quad NP : a$

S :  $f(a) \rightarrow NP : f \quad VP : a$

# Outline

---

- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:

->Articoli indeterminativi ->Nomi propri ->**Verbi transitivi**

->Cordinazione dei nomi ->Quantificatori universali

- NLTK
- Altri approcci alla semantica

# Problema con i verbi transitivi

---

“ama Francesca” funziona **ancora**?

Per i verbi transitivi:

ama <- V :  $\lambda y.\lambda x.\text{love}(x, y)$

Ma con i nomi propri:

Francesca <- NP:

$\lambda P.P(\text{francesca})$

quindi “ama Francesca”:

$\lambda y.\lambda x.\text{love}(x, y) @ \lambda P.P(\text{francesca}) \rightarrow (\lambda x.\text{love}(x, \lambda P.P(\text{francesca})))$



# Problema con i verbi transitivi

“ama Francesca” funziona **ancora**?

Per i verbi transitivi:

$\text{ama} \leftarrow V : \lambda y. \lambda x. \text{love}(x, y)$

Ma con i nomi propri:

$\text{Francesca} \leftarrow \text{NP}:$

$\lambda P.P(\text{francesca})$

quindi “ama Francesca”:

$\lambda y. \lambda x. \text{love}(x, y) @ \lambda P.P(\text{francesca}) \rightarrow (\lambda x. \text{love}(x, \lambda P.P(\text{francesca})))$

Type-raising

old type:  $e \rightarrow (e \rightarrow t)$

new type:  $((e \rightarrow t) \rightarrow t) \rightarrow (e \rightarrow t)$

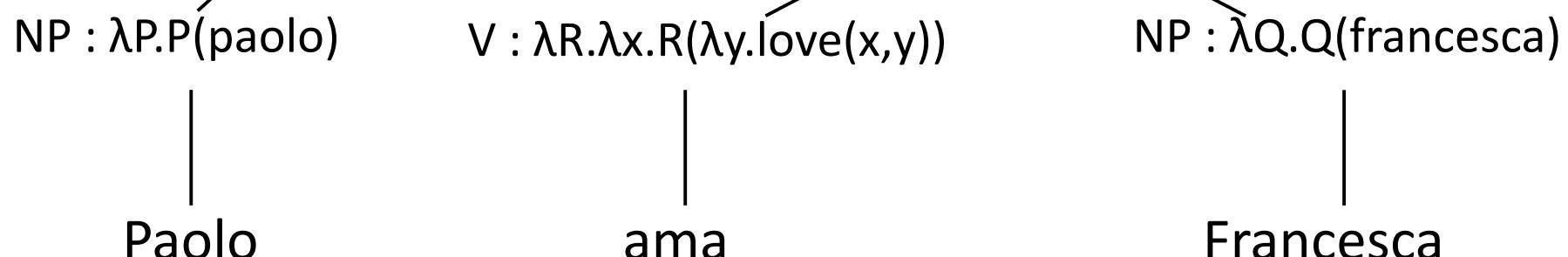
**Soluzione:** ancora type raising!

$\text{ama} \leftarrow V: \lambda R. \lambda x. R(\lambda y. \text{love}(x, y))$

# Paolo ama Francesca

$S : \lambda P.P(paolo)\lambda x.\text{love}(x,\text{francesca})$   
 $= \lambda x.\text{love}(x,\text{francesca})(paolo)$   
 $= \text{love}(\text{paolo},\text{francesca})$

$\text{VP: } \lambda R.\lambda x.R(\lambda y.\text{love}(x,y))(\lambda Q.Q(\text{francesca}))$   
 $= \lambda x.(\lambda Q.Q(\text{francesca}))(\lambda y.\text{love}(x,y))$   
 $= \lambda x.(\lambda y.\text{love}(x,y))(\text{francesca})$   
 $= \lambda x.\text{love}(x,\text{francesca})$



# Una semplicissima grammatica semantica (4)

nomi comuni	<i>uomo</i>	$\lambda x.\text{man}(x)$
nomi propri	<i>Paolo</i>	$\lambda P.P(\text{Paolo})$
verbi intr.	<i>corre</i>	$\lambda x.\text{run}(x)$
verbi tr.	<i>ama</i>	$\lambda R.\lambda x.R(\lambda y.\text{love}(x, y))$
articoli	<i>un</i>	$\lambda P.\lambda Q.\exists z(P(z) \wedge Q(z))$

## Punti chiave

- extra  $\lambda$  per gli NP
- astrazione sui predicati
- inversione di controllo: NP\_subj come funzione e VP come argomento

**Sistematicità**

# Outline

---

- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:
  - >Articoli indeterminativi ->Nomi propri ->Verbi transitivi
  - >**Cordinazione dei nomi** ->Quantificatori universali
- NLTK
- Altri approcci alla semantica

# Coordinazione dei nomi

---

- *Paolo e Francesca corrono*
  - $\text{run(Paolo)} \wedge \text{run(Francesca)}$

# Coordinazione dei nomi

---

- *Paolo e Francesca corrono*
  - $\text{run}(\text{Paolo}) \wedge \text{run}(\text{Francesca})$
- Aggiungiamo al lessico:
  - $\text{CC} : \lambda X. \lambda Y. \lambda R. (X(R) \wedge Y(R)) \rightarrow e$

# Paolo e Francesca corrono

$$\lambda R. (R(paolo) \wedge (R(francesca))(\lambda x. run(x))) \\ = \text{run(paolo)} \wedge \text{run(francesca)}$$

$$(\lambda Y. \lambda R. (R(paolo) \wedge Y(R))) (\lambda P. P(francesca)) \\ = \lambda R. (R(paolo) \wedge (\lambda P. P(francesca))(R)) \\ = \lambda R. (R(paolo) \wedge (R(francesca)))$$

$$(\lambda X. \lambda Y. \lambda R. (X(R) \wedge Y(R))) (\lambda P. P(paolo)) \\ = (\lambda Y. \lambda R. (\lambda P. P(paolo))(R) \wedge Y(R)) \\ = (\lambda Y. \lambda R. (R(paolo) \wedge Y(R)))$$

$\lambda P. P(paolo)$

Paolo

$\lambda X. \lambda Y. \lambda R. (X(R) \wedge Y(R))$

e

$\lambda P. P(francesca)$

Francesca

$\lambda x. run(x)$

corrono

# E per le altre coordinazioni?

---

- Paolo corre e vince
- Paolo mangia pane e cipolla
- Paolo corre e Francesca vede la gara

# Outline

---

- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:

->Articoli indeterminativi ->Nomi propri ->Verbi transitivi

->Cordinazione dei nomi ->**Quantificatori universali**

- NLTK
- Altri approcci alla semantica

# Scope ambiguity

---

*In this country, a woman gives birth every 15 minutes.*

# Scope ambiguity

In this country, **a**woman gives birth every 15 minutes.

Our job is to find that woman and stop her.

Groucho Marx

ambiguità pronome semantico

vs

"mango la piza con l'arancia / frutta"

è SINTACCIALMENTE ambiguo → ha 2 alberi

è SEMANTICAMENTE ambiguo  
↳ 1 albero, 2 latture

# Scope ambiguity

Every man loves @ woman

Lettura 1:

$$\forall x (\text{man}(x) \rightarrow \exists y (\text{woman}(y) \wedge \text{love}(x, y)))$$

entrambe  
condividono  
lo stesso  
albero  
sintattico

Lettura 2:

$$\exists y (\text{woman}(y) \forall x (\text{man}(x) \rightarrow \text{love}(x, y)))$$

$$\lambda Q \lambda P (\forall x (Q(x) \rightarrow P(x))) \rightarrow \text{Every}$$

il nostro alg. è deterministico, e dato lo stesso albero sint. e le stesse foglie,  
ci dà lo stesso significato; non permette di avere 2 letture

# CCG e semantica

Paolo

ama

NP

$(S \setminus NP) / NP$

P

$\lambda y \lambda x [love(x, y)]$

Francesca

NP

F

>

$S \setminus NP$

$\lambda x [love(x, F)]$

<

S

$love(P, F)$

# Avverbi: *Paolo ama Francesca dolcemente*

---

- **sweetly(love(P,F))** : second order!!!!
- **love(P,F,sweetly)**: e se ho più avverbi??

# Avverbi: *Paolo ama Francesca dolcemente*

---

- Soluzione: **reificare l'evento**, ovvero definire una variabile che identifica l'evento (Davidson)

$\exists e \ love(e, P, F) \wedge sweetly(e)$

- Si può generalizzare anche per gli argomenti:
  - neo-Davidsonian style (Parson) ->

$\exists e \ love(e) \wedge agent(e, P) \wedge patient(e, F)$

# *Paolo ama Francesca dolcemente*

---

**ESERCIZIO:** costruire la derivazione (albero e lambda-FoL termini) in stile Neo-Davidsoniano per:

**Paolo ama Francesca dolcemente**



$\exists e \ love(e) \wedge \text{agent}(e,P) \wedge \text{patient}(e,F) \wedge \text{sweetly}(e)$

# (Groningen & Parallel) Meaning Bank

---

- Annotazione completa
  - Sintassi: CCG
  - Semantica: Discourse Representation Theory (DRT), a formal theory of meaning developed by the philosopher of language Hans Kamp (Kamp, 1981; Kamp and Reyle, 1993)  
-> Equivalente a FoL.
- GMB: <http://gmb.let.rug.nl>
  - *A free semantically annotated corpus that anyone can edit!*
- PMB: <http://pmb.let.rug.nl>

# Outline

---

- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:

->Articoli indeterminativi ->Nomi propri ->Verbi transitivi

->Cordinazione dei nomi ->Quantificatori universali



- NLTK
- Altri approcci alla semantica

# nltk.sem [Garrette & Klein 2008]

---

nltk.sem package (Python):

- First-order logic & lambda calculus
- Theorem proving, model building, & model checking
- DRT & DRSSs
- Cooper storage, hole semantics, glue semantics
- Linear logic

<http://www.nltk.org/book/ch10.html>

# Paolo ama Francesca in NLTK

---

```
>>> from nltk import load_parser  
>>> parser = load_parser('./simple-sem-it.fcfg', trace=0)  
>>> sentence = 'Paolo ama Francesca'  
>>> tokens = sentence.split()  
>>> for tree in parser.parse(tokens):  
...     print(tree.label()['SEM'])  
  
...  
love(paolo,francesca)    mano
```

# simple-sem-it.fcfg

---

```
## Natural Language Toolkit: A simple example for Italian
## Author: Alessandro Mazzei <mazzei@di.unito.it>
## To see more complex example -> simple-sem.fcfg

% start S
#####
# Grammar Rules
#####

S[SEM = <?subj(?vp)>] -> NP[SEM=?subj] VP[SEM=?vp]
NP[SEM=?np] -> PropN[SEM=?np]
VP[SEM=<?v(?obj)>] -> V[SEM=?v] NP[SEM=?obj]
#####
# Lexical Rules
#####

PropN[SEM=<\P.P(paolo)>] -> 'Paolo'
PropN[SEM=<\P.P(francesca)>] -> 'Francesca'
V[SEM=<\X x.X(\y.love(x,y))>] -> 'ama'
```

# Un esempio più complesso in Inglese

---

```
>>> from nltk import load_parser  
>>> parser = load_parser('./simple-sem.fcfg', trace=0)  
>>> sentence = 'Angus gives a bone to every dog'  
>>> tokens = sentence.split()  
>>> for tree in parser.parse(tokens):  
...     print(tree.label()['SEM'])  
  
...  
all z2.(dog(z2) -> exists z1.(bone(z1) & give(angus,z1,z2)))
```

<http://www.nltk.org/howto/inference.html>

# Outline

---

- Chat-80
- Computational Semantics fundamentals
- $\lambda$  abstraction
- CS per:

->Articoli indeterminativi    ->Nomi propri ->Verbi transitivi

->Cordinazione dei nomi    ->Quantificatori universali

- NLTK
- Altri approcci alla semantica

- **AMR**

- Filler-slot

- **Semantic Grammars**



# 1. Semantica formale meno formale -> AMR

# Abstract Meaning Representation

---

<http://amr.isi.edu>

- *The AMR Bank is a set of English sentences paired with simple, readable semantic representations.*
- *AMRs are rooted, labeled graphs that are easy for people to read, and easy for programs to traverse.*
- *AMR aims to abstract away from syntactic idiosyncrasies.*
- *AMR makes extensive use of PropBank framesets (“bond investor” ->invest-01)*
- *AMR is agnostic about how we might want to derive meanings from strings, or vice-versa.*
- *AMR is heavily biased towards English. It is not an Interlingua.*

# AMR: A boy wants to go

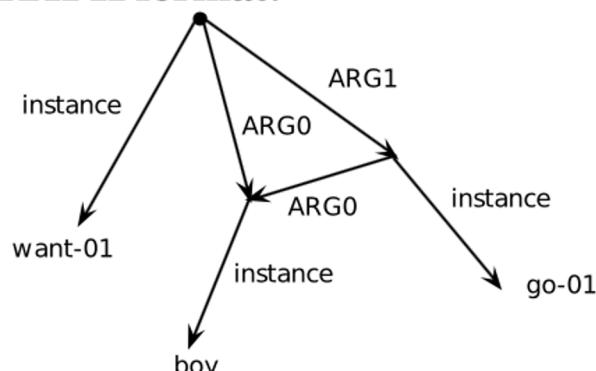
## LOGIC format:

$$\exists w, b, g: \\ \text{instance}(w, \text{want-01}) \wedge \text{instance}(g, \text{go-01}) \wedge \\ \text{instance}(b, \text{boy}) \wedge \text{arg0}(w, b) \wedge \\ \text{arg1}(w, g) \wedge \text{arg0}(g, b)$$

## AMR format (based on PENMAN):

```
(w / want-01
  :arg0 (b / boy)
  :arg1 (g / go-01
    :arg0 b))
```

## GRAPH format:



# AMR DOES NOT ...

---

- NO inflectional morphology for tense and number
- NO (many) universal quantifier
- NO distinguish between real events and hypothetical, future, or imagined ones
- J. Bos [http://www.mitpressjournals.org/doi/pdf/10.1162/COLI\\_a\\_00257](http://www.mitpressjournals.org/doi/pdf/10.1162/COLI_a_00257)
  - ... AMRs without recurrent variables can be translated into a decidable fragment of first-order logic.
  - ... at most one (wide-scope) universal quantifier can be used in standard AMRs

fine ~~X~~

• Siri & Co. usano però invece scacchiere:



## 2. Frame and Slot semantics

- nessun albero sintattico
- nessuna analisi sintattica/semantica

# Natural Language Understanding for conversational agents

---

- For speech dialogue systems, most common semantics is “Frame and slot semantics”.

# A frame example

Show me morning flights from Boston to SF on Tuesday.

SHOW:

FLIGHTS:

ORIGIN:

CITY: Boston

DATE: Tuesday

TIME: morning

DEST:

CITY: San Francisco

← "completa" dei formulari sulla  
base delle frasi pronunciate  
come fai? Tutto machine learning

I miei corpi mi dicono, tramite  
gli esempi, che "Boston" è  
una City, etc

filler slots semantics

# Sentence semantics

LIST\_FLIGHTS

ORIGIN

Show me flights from Boston

DESTINATION

DEPARTDATE

to San Francisco on Tuesday

DEPARTTIME

morning

CON: . è specifico di un dominio  
siaggia se a un chatbot aereo chiede altra  
. avere una semantica "flat", non  
ricorsiva, non può essere annidata.  
Non si possono usare fasi incapsolate

PRO: . con domini molto diversi  
ristoranti, appuntamenti, etc

# SIRI NLU: Condition-Action Rules

---

Active Ontology: relational network of concepts

- **data structures:** a meeting has

- a date and time
- a location,
- a topic
- a list of attendees

non ho più alberi,  
ma array

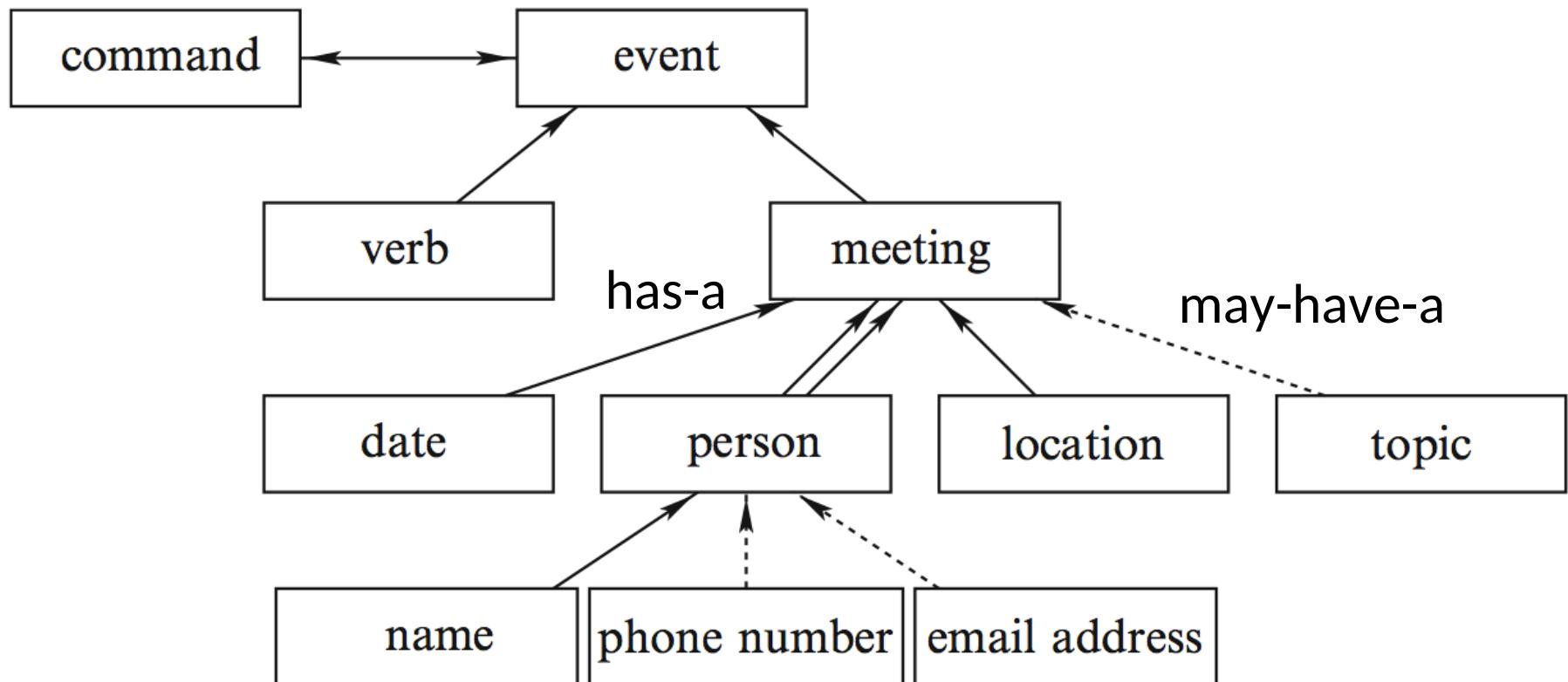
- **rule sets** that perform actions for concepts
  - the date concept turns string *Monday at 2pm* into date object  
`date(DAY,MONTH,YEAR,HOURS,MINUTES)`

# Rule set

---

- Collections of rules consisting of:
  - condition
  - action
- When user input is processed, facts added to store and
  - rule conditions are evaluated
  - relevant actions executed

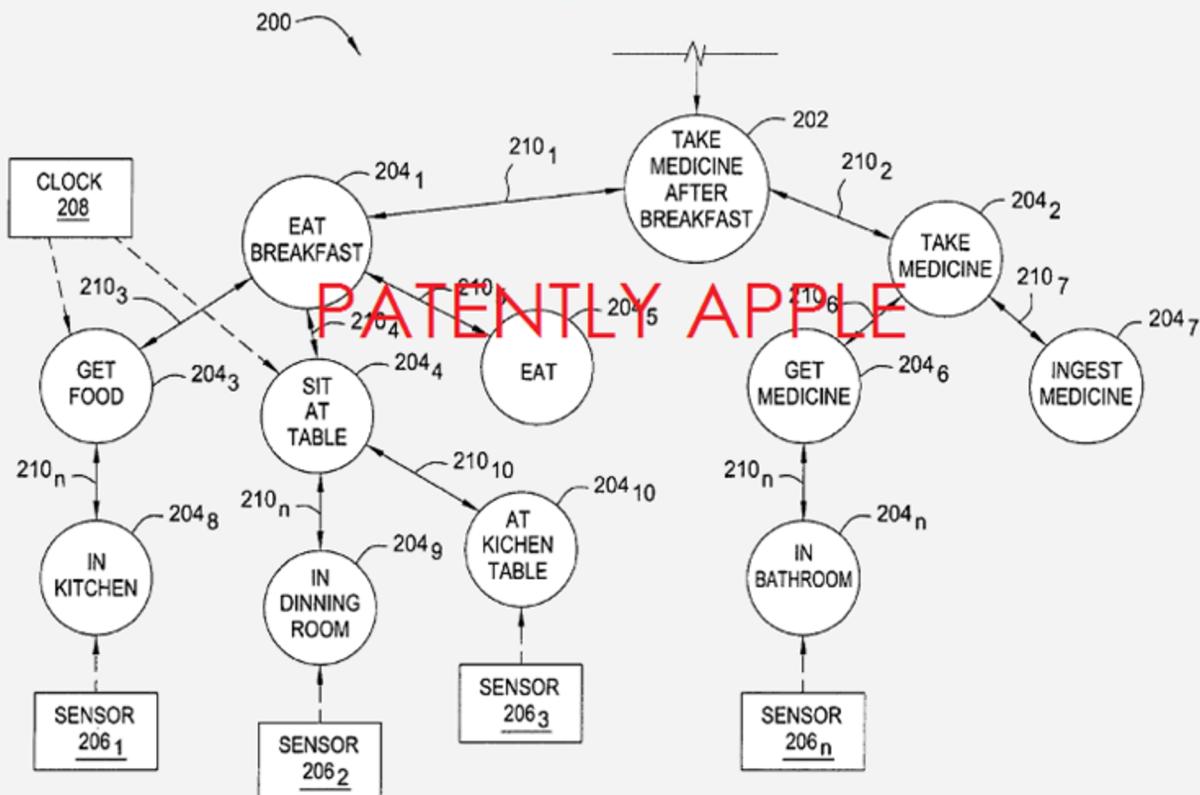
# Part of ontology for meeting task



Idea: Meeting Concept -> *if you don't yet have a location, ask for a location*

Apple Granted Patent for Advancements in Siri: Auto Reminder System

FIG. 2



# Improvements to the Rule-Based Approach

---

- Statistical classifiers to map words to semantic frame-filters
- Given a set of labeled sentences
  - “I want to fly to San Francisco on Tuesday”
    - Destination: SF
    - Depart-date: Tuesday
- Build a classifier to map from one to the author
- Requirements: Lots of labeled data

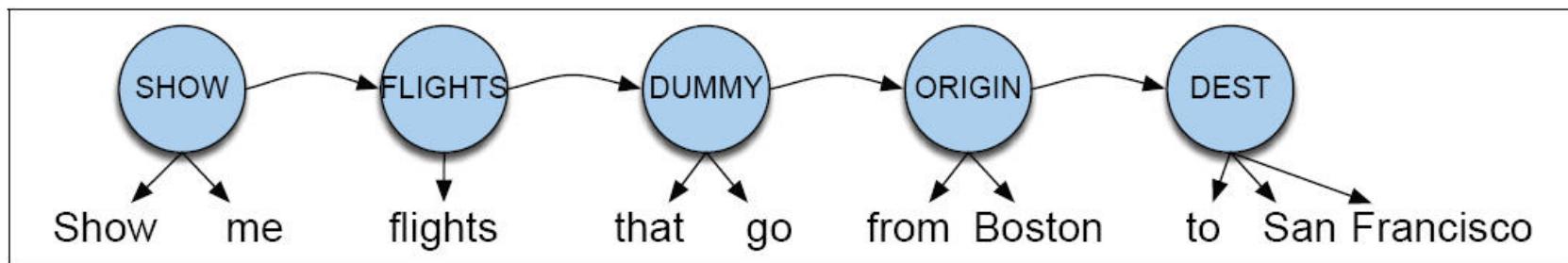
# HMMs for semantics

---

- Hidden units are slot names
  - ORIGIN
  - DESTCITY
  - DEPARTTIME
- Observations are word sequences
  - *on Tuesday*

# HMMs for semantics: Pieraccini et al. 1991

$P(W|C) = P(me|show, SHOW) \times P(show|SHOW) \times$   
 $P(flights|FLIGHTS) \times P(FLIGHTS|SHOW) \times$   
 $P(DUMMY|FLIGHTS) \dots$





### 3. Semantic grammars

# Semantic Grammars -> Semantic CF rules

---

CFG in which the LHS of rules is a semantic category:

- LIST -> show me | I want | can I see|...
- DEPARTTIME -> (after|around|before) HOUR
  - | morning | afternoon | evening
- HOUR -> one|two|three...|twelve (am|pm)
- FLIGHTS -> (a) flight|flights
- ORIGIN -> from CITY
- DESTINATION -> to CITY
- CITY -> Boston | San Francisco | Denver | Washington

# Tina parse tree with semantic rules: Seneff 1992

