Analisi Semantica del Linguaggio Naturale - Prima Parte: Introduzione alla analisi grammaticale in Prolog

Roberto Basili

Department of Enterprise Engineering
University of Roma, *Tor Vergata*
Via Della Ricerca Scientifica, 00133, Roma, ITALY
e-mail: basili@info.uniroma2.it

a.a. 2024-25

Semantica delle Lingue e Logica

Outline

Uso del Calcolo dei Predicati (o FOL) in NLP ed AI

Obbiettivi della FOL in NLP

- Definire una formalismo per rappresentare la semantica delle frasi di una lingua: forme logica
- Definire una sintassi per le regole di riscrittura che traducano le frasi di un alingua in forme logiche
- Supportare il processo di generazione della forma logica (*parsing*)
- Riusare la forma logica per ulteriori inferenze basate su tale rappresentazione logica

Il ruolo del FOL

Usare il FOL per esprimere i significati delle espr. linguistiche

Gianni corre $\rightarrow$ $corre(g)$

Gianni vede Michele $\rightarrow$ $vede(g,m)$

Gianni	`g`
Michele	`m`
corre	`{ x : corre(x) }`
vede	`{ <x,y> : vede(x,y) }`

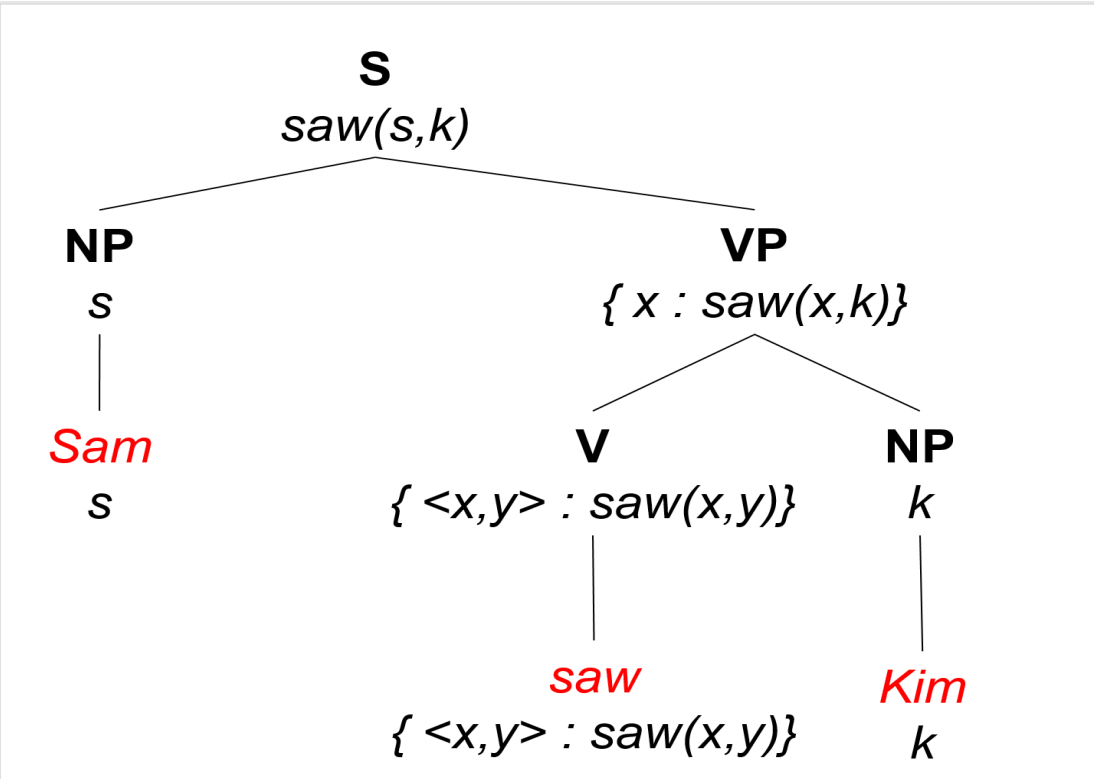
- Risultato: Fornire un sintassi per la *semantica*
- Qual'e' il processo di traduzione?

Composizionalita'

Il significato di una espressione e' una qualche funzione del significato delle sue parti e delle operazioni di combinazione (cioe' delle dipendenze/relazioni sintattiche). Ne segue che:

- Il significato di *Michele vede Gianni* e' una funzione del significato di *Michele* e di *vede Gianni*
- Il significato di *vede Gianni* e' una funzione del significato di *vede* e di *Gianni*
- L'interpretazione composizionale e' *ricorsiva* (rispetto alla sintassi)

Composizionalita'



Uso di FOL nella semantica composizionale

- La FOL ha una semantica composizionale: quindi va garantita una traduzione *verso* la FOL che sia composizionale
- Inoltre, il significato delle espressioni linguistiche deve essere messo in corrispondenza sistematica con il significato delle *parti* componenti
- Serve:
  - una traduzione semantica per ogni espressione di base o complessa
  - una operazione semantica per ogni regola sintattica di composizione

Il principio di composizionalita'

- Ogni regola sintattica puo' essere vista come una funzione che dati dei (o delle combinazioni di) morfemi produce una espressione in uscita
- Ogni regola sintattica R applicata ai morfemi $\alpha_1, \alpha_2, \ldots, \alpha_n$ produce una espressione in uscita ξ del tipo

$$\xi = R(\alpha_1, \ldots, \alpha_n)$$

- E' ragionevole assumere che ogni elemento atomico del linguaggio α (ad es. nomi) sia in corrispondenza, nel mondo, con una precisa entita' o proprieta' o relazione $sem(\alpha)$ (es. i nomi propri mappano verso individui/entita' nel mondo)
- Ne segue che ogni relazione R ammette una controparte semantica R' tale che:
se $\xi = R(\alpha_1, \ldots, \alpha_n)$ allora $sem(\xi) = R'(sem(\alpha_1), \ldots, sem(\alpha_n))$

Regole context-free e Logica

- Un Linguaggio è definito dall'insieme di regole di riscrittura che ne caratterizzano le strutture a partire da uno specifico non terminale S che costituisce la rappresentazione corrispondente. Ad esempio

$$S \rightarrow NP\ VP$$

  è una possibile descrizione delle frasi dichiarative soggetto-predicato che sono molto comuni in Italiano
- Esempio sono frasi come:
  "Gianni mangia", "Maria dorme di notte", La zia donò l'orologio a Mattia lo scorso anno, ..

Regole context-free e Logica (2)

- Le regole sintattiche possono essere formalizzate sia come regole di riscrittura (come da sempre i corsi di algoritmica le descrivono) ma anche, da un punto di vista logico, come assiomatizzazioni di una certa teoria sul linguaggio che si intende descrivere.
- Nell'esempio
  $S \rightarrow NP\ VP$
  può essere visto come un assioma del tipo

$$\forall x(\exists y,z \quad NP(y) \wedge VP(z) \wedge concat(y,z,x)) \rightarrow S(x) \qquad (1)$$

- La regola 1 descrive una coppia di sottofrasi y e z della x che riscritte nei non terminali NP e VP rispettivamente quaificano la x come frase (S) del linguaggio.

Parsing come deduzione in Logica

Le regole sintattiche come la (1) possono essere facilmente espresse dunque in Prolog, dove il formalismo dichiarativo proprio della logica coincide in modo diretto con le regole di riscrittura di una grammatica. Infatti la regola descritta nella Eq. (1)[1], corrisponde al seguente frammento Prolog

- `s(X) :- append(Y,Z,X), np(Y), vp(Z)`: dove
  - `append(Y,Z,X)` consente di enumerare tutte le coppie di sottosequenze `Y` e `Z` che concatenate ricoprono `X`, cioè la frase in input
  - le proprietà `np(Y)` e `vp(Z)` ricorsivamente garantiscono le `Y` e `Z` svolgano il ruolo di frase Soggetto e frase Verbale corrispondentemente.

[1] $\forall x(\exists y,z \quad NP(y) \wedge VP(z) \wedge concat(y,z,x)) \rightarrow S(x)$

Parsing come deduzione in Logica (2)

Le regole sintattiche come quella corrispondente al seguente frammento Prolog

```
s(X) :- append(Y,Z,X), np(Y), vp(Z)
```

danno luogo ad una processo di parsing grazie all'interprete Prolog. Questo proverà prima a decomporre la frase in ingresso[2] `X` in due sottofrasi `Y` e `Z`, poi applicherà ricorsivamente ad esse i criteri della riscrittura nei due non terminali `NP` e `VP` rispettivamente.

[2] Vedi predicato `append` a pagina successiva.

Digressione: il predicato `append/3`

Il seguente predicato Prolog

```
append([],X,X).
append([HY|TailY],Z, [HY|X]) :-
    append(TailY,Z,X).
```

può essere usato sia per calcolare la concatenazione delle due liste in ingresso, cioè

`?-append([a,b], [c], X).` con `X = [a,b,c]`

che per enumerare le due sottoliste di una lista data, come ad esempio in

```
?-append(Y, Z, [a,c]).
```

con `Y=[], Z=[a,c]; Y=[a], Z=[c]; Y=[a,c], Z=[]`

Parsing di un esempio: "Giovanni dorme'

Il seguente frammento di grammatica:

```
s(X) :- append(Y,Z,X), np(Y), vp(Z)
np(X) :- pn(X)
vp(X) :- v(X)
```

dovrebbe essere sufficiente per la frase

Giovanni dorme

L'unico requisito è descrivere come in dizionario, gli assiomi per rappresentare la categoria grammaticale delle due parole qui coinvolte di *''Giovanni'* e *'dorme'*. Esse sono un *nome proprio* ed un *verbo*, rispettivamente e quindi sono sufficienti i seguenti fatti Prolog:

```
pn(['Giovanni']).
v(['dorme']).
```

Parsing e riscrittura di un esempio: "Giovanni dorme'

Per riscrivere l'output della interpretazione della lista in ingresso, è possibile estendere le regole della grammatica aggiungendo un argomento in output, come segue:

```
s(X, s(NP,VP)) :-
   append(Y,Z,X), np(Y, NP), vp(Z, VP).
np(X, pn(X)) :- pn(X).
vp(X, v(X)) :- v(X).
```

Con queste estensione il parsing della frase *Giovanni dorme* si esegue nel modo seguente:

```
?-s(['Giovanni', 'dorme'], Tree).
```

con `Tree=s(pn(['Giovanni']), v(['dorme'])).`

che enfatizza la struttura ad albero della interpretazione.

Esercizio: completa la grammatica su frasi esempio

Estendere la grammatica e le regole lessicali sin qui descritte per elaborare con successo le seguenti frasi

```
sent(0,['Mario',mangia]).
sent(1,['Giovanni',scrisse,un,poema]).
sent(2,[il,libro,fu,scritto,da,due,autori]).
sent(3,[le,prime,sezioni,erano,noiose]).
sent(4,[le,ore,insonni,scorrevano,nella,notte,a,'Roma']).
sent(5,['Giovanna',di,notte,discorreva,del,
                                primo,capitolo,del,libro]).
```

Si aggiungano le regole per frasi verbali che includano il complemento oggetto (Frase 2) e gli altri complementi (Frasi 4 e 5), e si aggiungano le parole (verbi, nomi e aggettivi) al dizionario[3].

[3] Vedi implementazione della interfaccia utente nel file "Esercizio Parsing e Logica" distribuito con queste slide

Bibliografia

- Prolog Implementation: SWI-Prolog, URL: `https://www.swi-prolog.org/`
- Prolog Basics. `https://lpn.swi-prolog.org/lpnpage.php?pageid=online`
- Chapter 4. "*Further Topics in NL Analysis*", in Fernando C.N. Pereira and Stuart M. Shieber. "*Prolog and Natural-Language Analysis*", volume 10 of CSLI Lecture Notes. Chicago University Press, Stanford, 1987.
- *Intelligenza Artificiale*, S. J. Russel, P. Norvig, Prentice Hall Int., Chapter 22.3-22.8, 23, 1998.