Programmazione logica e linguaggi formali

Fondamenti di informatica, Lezione 26

Università di Catania, Facoltà di Scienze MM.FF.NN.

Corso di laurea in Informatica

Informatica

14 gennaio 2009

Sommario

elementi della programmazione logica

basata sulla deduzione in una **logica dei predicati ristretta:**

- **clausole Horn strettamente positive** $\gamma_1, \ldots, \gamma_n \rightarrow \alpha$
  - $\gamma_1, \ldots, \gamma_n, \alpha$: formule predicative atomiche (con variabili)
  - quantificazione universale implicita
- "scoperta da ***R. Kowalski*** nel 1974,
  implementata da ***A. Colmerauer*** nel 1973" ... $\Longrightarrow$ ***Prolog***
- **primitive:**

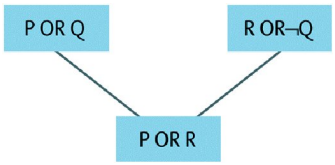
fatti: clausole Horn strettamente positive con $n = 0$
regole: clausole Horn strettamente positive con $n > 0$
programma: insieme di fatti e regole
***query*:** predicato, di solito con **variabili**
soluzione: **sostituzione** di valori per le variabili della *query* che ne rende la corrispondente istanza deducibile dal programma

- l'**ordine** di fatti e regole *dovrebbe* essere **irrilevante**

clausole Horn positive e principio di risoluzione

- **forma clausale** di un enunciato nella logica dei predicati:
  - congiunzione di un insieme (finito) di **clausole**, dove:
  - **clausola:** disgiunzione di un insieme finito di **letterali**, dove:
  - **letterale:** atomo (**positivo**)
    o negazione di atomo (**negativo**)

principio di risoluzione:



se **P** e **R** sono entrambi vuoti,
con la regola di risoluzione **si deduce la clausola vuota**,
ovvero l'**inconsistenza** della forma clausale data

clausola Horn (strettamente) positiva:
clausola in cui solo un atomo è positivo

è facile verificare l'equivalenza logica:

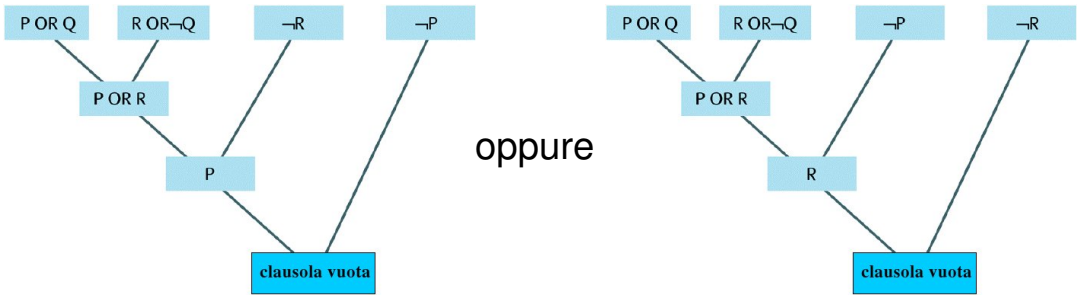
$$\alpha \vee \neg\gamma_1 \vee \ldots \vee \neg\gamma_n \equiv \gamma_1 \wedge \ldots \wedge \gamma_n \rightarrow \alpha$$

deduzione per refutazione

per dedurre il predicato **P** da una forma clausale **C**
si cerca di dedurre la clausola vuota da **C′ = C∧¬P**

esempio: **C = (P∨Q)∧(R∨¬Q)∧¬R**

- la deduzione per refutazione di **P** da **C** procede come in figura, usando il principio di risoluzione:



sostituzione e unificazione

le clausole Horn con variabili sono (implicitamente) universalmente quantificate: **C(X) ≡ ∀X C(X)**

- una **sostituzione** è una funzione da variabili in termini: **σ :V⟶T(V)**
- ogni sostituzione si estende naturalmente a termini e predicati, determinandone corrispondenti **istanze**

esempio: **σ: X↦0, Y↦X** **σ somma(X,Y,Y) = somma(0,X,X)**

- un **unificatore** di due predicati è una sostituzione che ne dà **istanze identiche** (sintatticamente)

esempio: **somma(X,0,X)** e **somma(0,Y,Y)** hanno l'unificatore **σ: X↦0, Y↦0**

- una **sostituzione chiusa** manda variabili in **termini chiusi**, cioè privi di variabili (come nell'esempio precedente)
- la programmazione logica si basa su risoluzione e unificazione, per determinare **soluzioni** di una ***query*** in un **programma**, ovvero sostituzioni chiuse che, applicate al predicato costituente la *query*, ne danno un'istanza deducibile dalle clausole Horn del programma

uso computazionale delle clausole Horn positive

convenzioni sintattiche ***Prolog*** **:**

- le **variabili** hanno iniziale maiuscola
- simboli di funzioni e di predicati hanno iniziale minuscola
- la clausola Horn $\gamma_1, \ldots, \gamma_n \rightarrow \alpha$ si scrive α **:-** $\gamma_1, \ldots, \gamma_n$ **.**

- il predicato binario infisso **is**, predefinito su termini aritmetici, è soddisfatto dall'identità di valutazione piuttosto che sintattica

esempio: un programma Prolog per il calcolo di **n!**, il **fattoriale di n:**

```
fact(0, 1) .
fact(N, M) :- N1 is N-1, fact(N1, Z), M is N*Z .
```

compilazione del programma ed esecuzione di una *query*:

```
?- [fact].
% fact compiled 0.00 sec, 1,004 bytes
?- fact(3,X).
X = 6
```

regole DCG in Prolog

- le **liste** sono una struttura dati predefinita in Prolog:

sintassi: **[]**, **[Head | List]**, e **[t_1, . . . , t_n]**

- si possono inoltre rappresentare regole di una grammatica libera, dette ***Definite Clause Grammar* (DCG) *rules*** o semplicemente ***grammar rules***, che vengono automaticamente tradotte in clausole Horn positive secondo una tecnica detta delle **liste-differenza**

sintassi: **t --> t_1, . . . , t_n .** (inoltre "**;**" separa alternative)

esempio 1: (i lessemi terminali sono in parentesi quadre)

```
frase          --> soggetto, predicato, ['.'].
soggetto       --> articolo, sostantivo.
predicato      --> verbo_intrans ;
                   verbo_trans, compl_oggetto.
compl_oggetto --> articolo, sostantivo.
sostantivo     --> [nutrice] ; [bimbo].
articolo       --> [il] ; [la] ; [un] ; [una].
verbo_intrans --> [dorme].
verbo_trans    --> [nutre].
```

vincoli sintattici con le regole DCG

si possono imporre **vincoli di concordanza** usando **variabili** e struttura di **termini** per simboli non terminali

esempio 2: così, rispetto all'esempio 1, si possono modificare le regole per `soggetto`, `compl_oggetto`, `articolo` e `sostantivo` come segue:

```
soggetto      --> articolo(Gen), sostantivo(Gen).
compl_oggetto --> articolo(Gen), sostantivo(Gen).
sostantivo(f) --> [nutrice].
sostantivo(m) --> [bimbo].
articolo(f)   --> [la] ; [una].
articolo(m)   --> [il] ; [un].
```

compilazione del programma ed esecuzione di *query*:

```
?- [nbg2].
% ngb2 compiled 0.00 sec, 3,096 bytes
?- phrase(frase, [il, bimbo, X, .]).
X = dorme ;
No
phrase(frase, [X, Y, nutre, il, bimbo, .]).
X = la
Y = nutrice
Yes
```

costruzione dell'analisi sintattica con regole DCG

- **variabili** e struttura di **termini** per simboli non terminali possono essere usati anche per la **costruzione dell'albero sintattico** di una data frase
- a tal fine le regole DCG della grammatica vanno modificate così che la *query* **phrase(frase(X), [lista_terminali]).** abbia come soluzione `X = t`, dove `t` sia l'albero sintattico di **lista_terminali**

esempio 3: ecco una modifica in tal senso rispetto all'esempio precedente:

```
frase(fs(S,P))             --> soggetto(S), predicato(P), ['.'].
soggetto(sg(A,S))          --> articolo(Gen,A), sostantivo(Gen,S).
predicato(pi(V))           --> verbo_intrans(V).
predicato(pt(V,O))         --> verbo_trans(V), compl_oggetto(O).
compl_oggetto(co(A,S))     --> articolo(Gen,A), sostantivo(Gen,S).
sostantivo(f,s(nutrice)) --> [nutrice].
sostantivo(m,s(bimbo))     --> [bimbo].
articolo(f,a(la))          --> [la].
articolo(f,a(una))         --> [una].
articolo(m,a(il))          --> [il].
articolo(m,a(un))          --> [un].
verbo_intrans(vi(dorme)) --> [dorme].
verbo_trans(vt(nutre))     --> [nutre].
```

automazione dell'analisi sintattica con Prolog DCG

la tecnica esemplificata sopra trasforma in modo relativamente meccanico una grammatica libera in una DCG Prolog che può essere compilata ed eseguita per ottenere l'analisi sintattica delle espressioni del linguaggio che essa genera

esempio: compilazione del programma ed esecuzione di *query*:

```
?- [nbg2p].
% ngb2p compiled 0.00 sec, 4,136 bytes
?- phrase(frase(X), [la, nutrice, nutre, il, bimbo, .]).
X = fs(sg(a(la), s(nutrice)), pt(vt(nutre), co(a(il),
s(bimbo)))) ;
No
?- phrase(frase(X), [la, nutrice, Y, .]).
X = fs(sg(a(la), s(nutrice)), pi(vi(dorme)))
Y = dorme ;
No
?- phrase(frase(X), [Y, Z, W, .]).
X = fs(sg(a(la), s(nutrice)), pi(vi(dorme)))
Y = la
Z = nutrice
W = dorme
Yes
```

esercizi: programmazione logica e DCG

1. La regola classica di deduzione logica detta *modus ponens* può essere espressa come segue: ***se*** **A** ***e*** **A**$\rightarrow$**B** ***allora*** **B**. Mostrare che questa regola è equivalente a un caso particolare del principio di risoluzione.

Suggerimento: Tener conto dell'equivalenza logica **A**$\rightarrow$**B** $\equiv$ **B**$\vee\neg$**A**.

2. Estendere la grammatica dell'esempio illustrato a lezione con regole DCG che ammettano la congiunzione di frasi come frase del linguaggio
3. **Automazione della costruzione di analizzatori sintattici**

   Le ***Prolog*** **DCG** ***rules*** non sono trattate nel libro di testo adottato. Una rapida introduzione all'argomento è disponibile nel sito del corso