Introduzione al PROLOG

Laboratorio di apprendimento logico-matematico

Rapporti con la logica

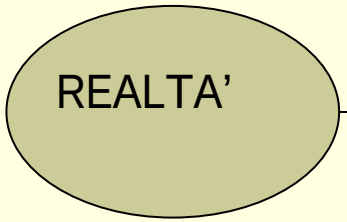
Ogni sistema “intelligente” si basa essenzialmente sulla capacità di:

- riconoscere premesse date;
- dedurre conclusioni attraverso una rete di inferenze logiche.

Quanto asserito induce a pensare che la realtà possa essere descritta attraverso una formalizzazione di **fatti** e **relazioni** tra essi.

In tale ipotesi il problema diventa quello del saper interrogare una **base di conoscenze (BDD)**.

Schema formalizzato per descrivere la realtà



fatti (asserzioni)

regole (clausole)

descrittore

<fatto>: <predicato>(argomento, …, argomento)

<regola>: relazioni condizionate tra oggetti

La soluzione si realizza attraverso la definizione di quesiti semplici o complessi pertinenti con la realtà descritta.

Il soddisfacimento di quesiti posti garantisce il buon esito della ricerca nella BDD.

PROLOG: Programmazione logica

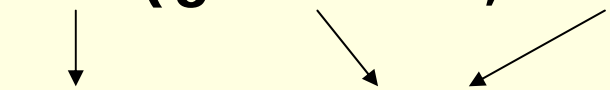
Il Prolog è un linguaggio artificiale caratterizzato essenzialmente dal:

- dichiarare fatti
- definire regole
- porre domande su fatti e relazioni

Esempio: **`F => Giovanni ama Maria`**

Questo fatto consiste di due oggetti detti “Giovanni” e “Maria” legati dalla relazione “ama”. Il fatto F in Prolog viene formalizzato in:

```
ama(giovanni,maria).
```



Ambiente PROLOG

In ambiente Prolog occorre:

U Definire l'universo degli oggetti

BDD Memorizzare i fatti e le asserzioni

Logica Definire le regole inferenziali

?? Richiedere informazioni coerenti con l'universo degli oggetti

Il linguaggio Prolog - introduzione

Il Prolog è un linguaggio che si presta a risolvere problemi che riguardano oggetti e loro relazioni.

Presenteremo il Prolog attraverso semplici esempi non curando molto in dettaglio il formalismo, ma cercando di enfatizzare gli schemi logici o di ragionamento che esso attiva per risolvere quesiti più o meno complessi.

La “scrittura” in Prolog consiste essenzialmente nel:

- DICHIARARE FATTI
- DEFINIRE REGOLE
- PORRE DOMANDE SU FATTI E RELAZIONI

Il linguaggio Prolog - introduzione

I fatti e le relazioni sono legati da “inferenze logiche” che ne costituiscono vere e proprie reti semantiche.

I fatti

Supponiamo di affermare:

`F => Giovanni ama Maria`

Questo fatto consiste di due “oggetti” detti “Giovanni” e “Maria” legati dalla relazione “ama”.

Il fatto F in Prolog va tradotto in:

`ama(giovanni,maria).`

secondo le seguenti regole…

Il linguaggio Prolog - introduzione

a) nomi e oggetti devono iniziare con lettere minuscole;
b) la relazione deve essere scritta prima degli oggetti, comunque raggruppati tra parentesi;
c) il punto deve concludere la descrizione di un fatto.

Oss.1: Nella descrizione dei fatti occorre essere coerenti con ciò che si desidera descrivere.

Oss.2: I nomi degli oggetti compresi tra parentesi tonde si chiamano "argomenti": la relazione è detta "predicato".

L'insieme dei fatti in Prolog prende il nome di Base dei Dati (BDD).

Il linguaggio Prolog - introduzione

Le domande

Consideriamo la seguente domanda:

D => Mara possiede il libro?

In Prolog la domanda D viene formalizzata in:

possiede(mara, libro)

Oss.3: Prolog alla domanda posta esplora la sua BASE DI DATI e va alla ricerca di un fatto coincidente con quanto esplicitato dalla domanda.

La risposta affermativa si avrà nel caso in cui il quesito posto venga soddisfatto.

Il linguaggio Prolog - introduzione

ESEMPIO

Definiamo una BASE DI DATI in cui siano acquisiti i seguenti fatti:

```
piace(giovanni,frutta).
piace(giovanni,mara).
piace(giovanni,libro).
piace(mara,libro).
```

Poniamo le nostre domande ed esaminiamo il comportamento di Prolog.

Il linguaggio Prolog - introduzione

Domanda	PROLOG
`piace(giovanni,vino)`	**`False`** (no)
`piace(mara,giovanni)`	**`False`** (no)
`piace(giovanni,libro)`	**`True`** (si)

In base ai fatti conosciuti da Prolog nei primi due casi il quesito risulta non soddisfatto, nel terzo caso ha esito positivo.

Le variabili

Può accadere che un predicato operi su più oggetti; o, in altri termini, occorre indagare quali oggetti soddisfino ad una determinata condizione.

Riferiamoci alla nostra BDD precedentemente definita e cerchiamo di risolvere il quesito:

```
D => Quali oggetti piacciono a Giovanni?
```

Per risolvere il quesito proposto occorre introdurre il concetto di variabile, secondo le seguenti convenzioni:

a) ogni variabile inizia con lettera maiuscola;
b) una variabile può essere istanziata o no.

Una variabile si dice *istanziata* se ad essa il Prolog associa un oggetto.

Le variabili

La domanda D viene formalizzata in Prolog in:

`piace(giovanni,X)`

Il Prolog soddisfa il quesito posto esplorando la sua BASE DI DATI e cercando se esistono fatti coincidenti con il quesito proposto.

a) La variabile X al momento della domanda non è istanziata.

b) Prolog esplora la BDD ricercando fatti che presentano lo stesso predicato (“piace”).

c) Se la ricerca ha successo, la variabile X viene istanziata dal secondo argomento del fatto trovato e Prolog visualizza la variabile X.

d) Prolog provvede a segnare con un puntatore la posizione in cui è stato trovato il fatto coincidente nella BDD.

e) Vengono effettuate tutte le ricerche possibili, segnalando i successivi valori di X.

Le variabili

- Esaminiamo la domanda: **`D => piace(giovanni,X)`**

In riferimento alla BDD prima definita il Prolog è in grado di rispondere al quesito proposto:

1a risposta => **`X = frutta`**
2a risposta => **`X = mara`**
3a risposta => **`X = libro`**
(fine ricerca)

La variabile X è stata istanziata durante l'esplorazione della BDD dagli oggetti in relazione con il predicato (piace).

Naturalmente tali oggetti costituiscono i soli fatti che Prolog riconosce in fase di esplorazione.

La conoscenza, però, può essere arricchita inserendo nella BDD altri fatti.

Oss. La coerenza nella descrizione dei fatti che costituiscono la BDD è fondamentale perché si possa riconoscere al Prolog un comportamento logico e consequenziale.

Qualche complicazione

Nel porre i quesiti può accadere che il livello di complessità delle relazioni di cui si voglia verificare la "verità" può richiedere il porre domande del tipo:

```
D => Giovanni e Mara si piacciono?
```

Una tale proposizione dal punto di vista formale chiede il soddisfacimento di due condizioni, cioè:

1. Verificare se a Giovanni piace Mara
2. Verificare se a Mara piace Giovanni

Il problema consiste cioè nel rispondere a due domande successive; in termini Prolog il quesito richiede di soddisfare due domande (GOAL) diverse ma in modo congiunto, e sarà:

```
piace(giovanni,mara), piace(mara,giovanni)
```

GOAL congiunti

Applichiamo il concetto di soddisfacimento congiunto alla BDD che andiamo a definire:

```
piace(giovanni,frutta).
piace(giovanni,mara).
piace(giovanni,libro).
piace(mara,libro).
```

Volendo sapere quali sono le cose che piacciono sia a Giovanni che a Mara, il quesito Prolog sarà:

```
piace(giovanni,X), piace(mara,X)
```

GOAL congiunti

Esaminiamo la proposizione e osserviamo che sono identificabili due GOAL, cioè:

```
piace(giovanni,X).
piace(mara,X).
```

I due GOAL (obiettivi) sono legati dalla congiunzione “e”, rappresentata in modo formale dal simbolo “,”.

GOAL congiunti

Come il Prolog risolve il quesito…

1. Prolog esamina la proposizione e cerca di risolvere il primo GOAL.
2. Prolog esplora la BDD e marca la posizione in cui si trova il primo GOAL.
3. Prolog riparte nella ricerca per soddisfare il secondo GOAL.

Se anche il secondo GOAL viene soddisfatto, allora Prolog marca nella BDD la posizione e fornisce il risultato. Ogni GOAL soddisfatto mantiene (ricorda) la posizione attraverso un marker (puntatore).

Il comportamento del Prolog

1. Prolog fissa il primo GOAL (la variabile X non è istanziata);
   esplora la BDD;
   il primo GOAL viene soddisfatto e la variabile X viene istanziata dall'oggetto "frutta" (Prolog segna la posizione in cui è stato trovato il "fatto" nella BDD).

Il comportamento del Prolog

2. Prolog esplora la BDD alla ricerca del secondo GOAL che si presenta con la variabile istanziata a “frutta”. Come è possibile osservare, nella BDD non esiste tale asserto e quindi il GOAL fallisce. Prolog, partendo dal successivo asserto tenta di soddisfare il GOAL precedente rendendo la variabile X non istanziata.

Il comportamento del Prolog

3. Prolog esplora la BDD dal fatto: **`piace(giovanni,mara).`**
   La variabile X viene istanziata a "mara".
   Il GOAL viene soddisfatto e Prolog marca tale posizione nella BDD.

Il comportamento di Prolog

4. Prolog cerca di soddisfare il secondo GOAL ricercando questa volta:
   `piace(mara,mara).`
   Il GOAL non viene soddisfatto e si ricomincia il processo.

Il comportamento di Prolog

5. Prolog riprende l'esplorazione del primo GOAL, e cerca l'esistenza del fatto: **piace(giovanni,libro).**
   Il GOAL viene soddisfatto alla parola "libro", Prolog marca la posizione del ritrovamento nella BDD e passa al secondo GOAL, cioè **piace(mara,X)**, in cui X è istanziata dall'oggetto "libro". Questa volta il secondo GOAL è soddisfatto e il Prolog comunica qual è l'oggetto che piace ad entrambi i protagonisti della ricerca.

Il backtracking

E’ il processo secondo il quale il Prolog cerca di soddisfare il GOAL definito effettuando una ricerca dall’alto verso il basso e da sinistra a destra, secondo uno schema informativo “ad albero”.

Se, durante la percorrenza dell’albero, si giunge ad un punto morto, Prolog retrocede (*backtracking*) nella lista delle clausole per trovare un altro percorso che soddisfi il GOAL.

E’ evidente che nei processi di backtracking si producono numerosi tentativi di ricerca dipendenti dalle regole e dai fatti definiti nella BDD.

Le regole

Nel Prolog molto spesso occorre esprimere concetti e regole per consentire interpretazioni più complesse di "fatti" definiti nella BASE DEI DATI.

Se voglio esprimere il concetto che **"Giovanni ama tutte le persone"** posso fare una lista nominativa delle persone che Giovanni ama; ciò probabilmente non è applicabile quando la lista delle persone amate da Giovanni è troppo lunga.

Come si può esprimere diversamente il quesito? Proviamo a formalizzare meglio il quesito posto. Si potrebbe esprimere lo stesso concetto dicendo: **"Giovanni ama qualsiasi oggetto che sia una persona"**, oppure:

Giovanni ama X se X è una persona

Invero tra le due proposizioni che costituiscono la formula si crea una certa dipendenza. In questi casi occorre definire delle "REGOLE" che mettono in relazione fatti e circostanze che descrivono un evento da osservare.

Le regole

Proviamo a definire la seguente relazione di parentela: “X è fratello di Y”.

```
REGOLA→X è fratello di Y se:
      - X è di sesso maschile,
      - X ed Y hanno gli stessi genitori.
```

Nella definizione appena formulata occorre osservare che le variabili X ed Y rappresentano lo stesso oggetto, cioè appartengono all’insieme delle “persone”: ciò è importante perché si possa dare senso alle regole formulate.

Sarebbe strano pensare che la variabile X rappresentasse un libro!

In Prolog una regola è costituita da un antecedente (testa) e un conseguente (corpo o coda).

Le regole

Consideriamo l'affermazione:

Giovanni ama qualcuno se questi ama il vino

La testa della regola ("Giovanni ama qualcuno") ha lo scopo di chiarire quale fatto la regola deve definire.

Formalizzando in termini Prolog possiamo scrivere:

```
ama(giovanni,X):-ama(X,vino)
```

Il corpo della regola descrive la congiunzione di fatti che devono essere soddisfatti perché sia vera la testa.

Le regole

Invero possiamo rendere più selettiva la regola se asseriamo:

```
ama(giovanni,X):-
ama(X,vino),ama(X,libro)
```

La regola appena formulata è di tipo *condizionale*; il simbolo `:-` sta a denotare il condizionale (se).

Osserviamo che l'unica variabile che verrà istanziata dal verificarsi di certi fatti è rappresentata dalla X.

Essa sarà istanziata solo nel caso in cui si verifichino le condizioni specificate nel corpo della regola.

Esempio di definizione di regole

Supponiamo di creare una BASE DI DATI in cui siano definiti alcuni fatti (clausole) riguardanti i familiari della regina Vittoria:

```
maschio(albert).
maschio(edward).
femmina(alice).
femmina(victoria).
genitori(edward,victoria,albert).
genitori(alice,victoria,albert).
```

Definiamo le variabili:

X→(membro della famiglia)
M→(individua l'oggetto "madre")
P→(individua l'oggetto "padre")

Definiamo i predicati:

maschio(X) (ad un solo argomento)
femmina(X)(")
genitori(X,M,P) (a tre argomenti, intendendo che M e P sono i genitori di X)

Esempio di definizione di regole

Si richiede di applicare una regola che definisca quali sono le relazioni perché X sia sorella di Y.

X è sorella di Y se:

- X è di sesso femminile
- X ha madre M e padre P
- Y ha la stessa madre M e padre P di X

Formalizziamo in termini Prolog:

```
sorella_di(X,Y) :-
    femmina(X),
    genitori(X,M,P),
    genitori(Y,M,P).
```

Nota: una clausola identifica fatti e regole relative a un predicato.

Esempio di definizione di regole

Analizziamo il quesito precedente.

Osserviamo che la testa della regola è rappresentata da **`sorella_di(X,Y)`**; tale asserzione sarà vera se tutti i GOAL del corpo della regola verranno soddisfatti nel corso dell'esplorazione della BDD. Poniamo la domanda in Prolog:

`sorella_di(alice,edward)`

Il quesito posto consiste per l'appunto nel verificare se, in base alle conoscenze presenti nella BDD, risulta che "Alice è sorella di Edward".

Come procede il Prolog?

1. La domanda posta coincide con l'unica regola definita: **`sorella_di(X,Y)`**; ne consegue che alle variabili X ed Y corrispondono rispettivamente gli oggetti "alice" e "edward" ai quali esse vengono istanziate. Nello stesso tempo viene marcata la regola e Prolog tenta di soddisfare i GOAL presenti nel corpo della regola.

Come procede il Prolog?

2. Il primo GOAL è **`femmina(alice)`**. Tale GOAL è vero in base ai fatti definiti nella BDD: Prolog segna nella BDD la posizione in cui esso risulta soddisfatto. A questo punto cerca di soddisfare il GOAL successivo, cioè: **`genitori(alice,M,P)`**. Nella BDD viene riportato un fatto che è associabile al GOAL da ricercare. Infatti Prolog trova: **`genitori(alice,victoria,albert)`**. Ciò consente di istanziare le variabili M e P rispettivamente agli oggetti “victoria” e “albert”, e Prolog marca la posizione nella BDD.

Come procede il Prolog?

3. Prolog continua la sua esplorazione e cerca nella BDD il GOAL:
   `genitori(edward,victoria,albert)`.
   Si ricordi che la variabile Y è stata istanziata all'oggetto "edward". Il GOAL viene soddisfatto dalla presenza nella BDD del fatto:
   `genitori(edward,victoria,albert)`.
   Ormai tutti i goal presenti nella regola sono stati soddisfatti e quindi anche la domanda posta:
   `sorella_di(alice,edward)`
   viene soddisfatta; ne consegue che il quesito posta risulta essere vero e quindi Prolog comunica quanto verificato attraverso la risposta (True).

Esercizi risolti – Enigma giallo

Il quesito che ci poniamo consiste nello scoprire se il nostro amico Tony può rubare qualcosa a qualcuno.

Supponiamo di definire alcuni fatti nella BDD:

1. `ladro(tony).`
2. `possiede(mara,denaro).`
3. `possiede(mara,libro).`
4. `piace(tony,X):-possiede(X,denaro).`
5. `puo_rubare(X,Y):-ladro(X), piace(X,Y).`

Il quesito posto può essere espresso in Prolog come:

`puo_rubare(tony,X)` (a chi può rubare Tony?)

Esercizi risolti – Enigma giallo

Schema di ragionamento:

a) perché Tony possa rubare qualcosa a qualcuno occorre che sia un ladro: il fatto (1) conferma tale ipotesi;

b) deve esistere qualcosa che posa piacere a Tony e che intenda rubare. Il fatto (4) afferma che a Tony interessa chi possiede denaro;

c) deve esistere qualcuno che possiede denaro. Il fatto (2) afferma che Mara possiede denaro. Quindi a Tony interessa Mara.

Concludendo: esistono tutte le condizioni perché Tony possa rubare; infatti:

- Tony è un ladro;
- a Tony interessa il denaro posseduto da Mara.

Esercizi risolti – Cesare e Marco

Il pompeiano Marco, come tutti gli umani, era fedele a qualcuno.

Nel periodo in cui Marco visse era imperatore cesare; i romani di allora erano divisi in due fazioni: quelli fedeli a Cesare e quelli che lo odiavano.

Poiché è noto che la gente cerca di assassinare gli imperatori cui non è fedele e tenendo conto che Marco tentò di assassinare Cesare, ci si chiede: `Marco era fedele a Cesare?`

Il quesito posto deve trovare soddisfacimento in uno schema di ragionamento (rete inferenziale) che dobbiamo formalizzare attraverso fatti e regole, basati sul dominio “persona”.

In TurboProlog avremo:

Esercizi risolti – Cesare e Marco

```
domains
  persona=symbol
predicates
  umano(persona)
  pompeiano(persona)
  imperatore(persona)
  non_fedele_a(persona,persona)
  fedele_a(persona,persona)
  odia(persona,persona)
  tenta_uccidere(persona,persona)
```

Esercizi risolti – Cesare e Marco

```
clauses
  umano(marco).
  pompeiano(marco).
  imperatore(cesare).
  tenta_uccidere(marco,cesare).
  fedele_a(X,Y):-umano(X), umano(Y).
  non_fedele_a(X,Y):-umano(X),
     imperatore(Y), tenta_uccidere(X,Y).
  odia(X,Y):-umano(X), imperatore(Y).
```

Esercizi risolti – Cesare e Marco

Osservazioni.

- Se ci si chiede se c'è un rapporto di non fedeltà tra due persone, la risposta che Prolog fornisce è: Marco - Cesare.
- Se ci si chiede se Marco è fedele a Cesare, la risposta fornita da Prolog è: False.
- Se ci si chiede se qualcuno odia Cesare, la risposta che Prolog fornisce è: Marco.

Quindi Prolog riesce in modo sintetico a dare indicazioni sulla verità o falsità di un quesito posto; ma in molti casi è in grado di associare a variabili "oggetti" del dominio.