Lezione15

Table of contents

Il Linguaggio dei Termini

È possibile realizzare sistema d'Intelligenza Artificiale, con un approccio basato su ragionamento. È stato detto che uno dei 4 modi per realizzare una AI, è quello di considerare il pensiero razionale e poi il comportamento razionale, facendo riferimento agli agenti che agiscono sul mondo a tentativo di arrivare a propri obbiettivi.

Prima di ciò, va parlato di come si affronta il ragionamento, i dati usati per la *programmazione logica* saranno non altro che deduzioni logiche; vedremo come costruire *procedure* usando queste deduzioni.

Il linguaggio dei termini è un insieme, per tutti i linguaggi, che si va a costruire utilizzando un insieme A chiamato *insieme dei simboli atomici* (insieme degli atomi) e un insieme V chiamato *insieme dei simboli di variabile* (insieme di variabili).
Questi due insiemi, supposti disgiunti, possono costruire un *insieme dei termini* (linguaggio dei termini) T di primo ordine, termini che garantiscono la caratteristica della logica di primo ordine. Descriviamo una grammatica, dicendo quando un certo set di simboli appartiene o meno al linguaggio:

- un qualsiasi simbolo di variabile $v \in V$ è un termine;
- un qualsiasi atomo $a \in A$ è un termine;
- se $f \in A$ è un simbolo atomico, considerando $k \geq 1 \in \mathbb{N}_+$ e un insieme di k termini $\{t_i\}_{i=1}^{k}$, allora scrivendo $f(t_1, t_2, \dots, t_k)$, con "," e "()" non appartenenti ne ad A ne a V, stiamo costruendo un termine che prende il nome di simbolo di funzione;
- nient'altro è un termine.

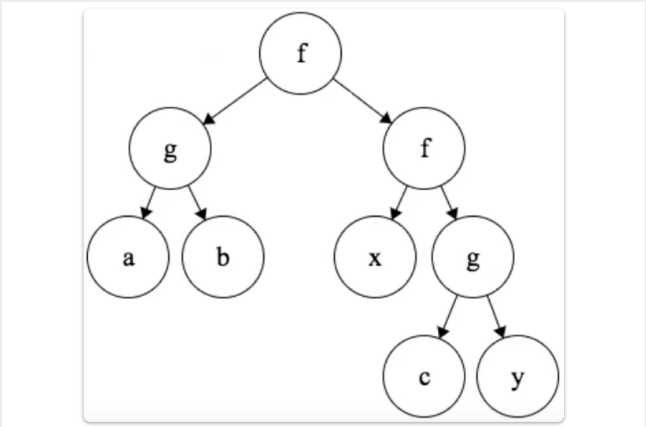
Il linguaggio dei termini si appoggia quindi all'alfabeto improprio $A \cup V$ e si basa sulla seguente grammatica non contestuale:

$$\begin{aligned} Term &\rightarrow Variable \mid Atom \mid Atom(Arguments) \\ Arguments &\rightarrow Term \mid Term, Arguments \\ Variable &\rightarrow \{v \in V\} \\ Atom &\rightarrow \{a \in A\} \end{aligned}$$

Un simbolo *non strutturato* è uno che non può essere spezzato, che non ha una struttura interna, e che quindi è un atomo o una variabile. Tutti gli altri simboli, formati da un atomo e le parentesi, vengono detti simboli *strutturati*: l'atomo più a sinistra viene detto termine e non può essere una variabile; il numero di argomenti si dice *arità*.

Esempi usando $A = \{a, b, f, g\}$ e $V = \{x, y\}$:

- a e x presi da soli sono termini;
- siccome f è un atomo, $f(a), f(x), f(a, f(x)), f(g(a), f(g(y, b)))$ sono termini, notando che f è stato utilizzato con arità 1 e 2 e che gli unici simboli su cui è stata applicata la notazione funzionale sono f e g.



Albero desritto dal termine $f(g(a,b),f(x,g(c,y)))$

Quello che stiamo descrivendo non sono altro che alberi.
Nella programmazione logica contemporanea, si utilizza questa nomenclatura: alla fine, si è capito che logico e funzionale, sono tipologie di approcci alla programmazione in cui i tipi dato elementari che vengono manipolati, sono alberi.
I tipi primitivi che vengono presi in matematica, prendono la forma di numeri, perché altro non sono: le stringhe, i caratteri, i riferimenti sono tutti numeri.

Dato un termine $t \in T$, elemento dell'insieme dei termini costruito su un certo insieme di atomi, $\texttt{vars}(t)$ contiene l'insieme delle variabili presenti in t. $\texttt{vars}(t)$ dell'albero in immagine sopra, riguarda le due variabili $\{x,y\}$ e quindi è uguale 2 (per caso coincide con V). Il caso più interessante è quando un termine non contiene variabili $\texttt{vars}(t) = \emptyset$: se così è, questo viene detto *ground*; non ha punti da approfondire.

Nei linguaggi che manipolano queste tipologie di dati, le costanti sono tutti termini ground perché riusciamo a costruire delle costanti strutturate: dire "se non uso mai $()$ allora è una costante", è dire il falso.

Dato un insieme di atomi non vuoto $A \neq \emptyset$, l'insieme dei termini che otteniamo utilizzando un insieme di variabili vuoto $V = \emptyset$, viene detto **universo di Herbrand**: insieme di tutte le costanti che potremmo andare a manipolare all'interno del nostro linguaggio di programmazione.

Due termini $t_1, t_2 \in T$ vengono detti *sintatticamente equivalenti* se sono lo stesso elemento di T.

Definiamo ora convenzioni per scrivere in modo più semplice, l'insieme dei termini.

1. Il simbolo "." viene tipicamente utilizzato per costruire termini strutturati formati da 2 argomenti.

   Quando il simbolo atomico che utilizziamo per costruire il termine strutturato è ".", anziché metterlo prima e usare "$()$" con "," per separare, mettiamo direttamente il "." in mezzo. Per esempio, invece che scrivere $.(a,b)$, scriviamo $a.b$.

2. Invece che utilizzare un insieme di simboli atomici qualsiasi, oltre al simbolo atomico speciale ".", aggiungiamo all'insieme dei simboli atomici, uno scritto come "$[\,]$".

Facendo ciò, possiamo permetterci di utilizzare la **sintassi delle liste**, un modo semplificato per costruite termini che utilizzano i simboli atomici appena descritti.

1. Un termine $[\,]$ si dice lista vuota.
2. $[t_1, t_2, \ldots, t_n]$ equivale a $.(t_1, .(t_2, \ldots, .(t_n, [\,]) \cdots))$ e si dice lista enumerata di n argomenti.
3. Scrivere $[h|r]$ equivale a $.(h,r)$, termine formato da una testa della lista h e un resto della lista r. Il secondo argomento è una variabile che può essere sostituita.
4. $[t_1, t_2, \ldots, t_n|r]$ equivale a $.(t_1, .(t_2, \ldots, .(t_n, r) \cdots))$.

Altra sintassi che utilizziamo per costruire dati che non sono altro che termini, è la cosiddetta **sintassi degli operatori**. Gli *operatori* sono simboli che trattiamo con sintassi speciale: il "." può essere visto come un operatore che però ha una sintassi speciale.

1. L'atomo da utilizzare per indicare l'operatore, può essere un $+$, un $-$, o altro.
2. La precedenza dell'operatore rispetto ad altri, mediante un indice naturale positivo.
3. Il tipo di operatore, mediante una delle sette tipologie ammesse
    1. fx, fy per gli operatori *unari prefissi*;
    2. xfx, xfy, yfx per gli operatori *binari infissi*;
    3. xf, yf per gli operatori *unari postfissi*.

Vediamo degli esempi:

- $(200, fy, -), (200, fy, +)$
    - fy descrive operatore prefisso, siccome f è il simbolo di operatore e y è qualsiasi espressione/termine che se ha operatore ha indice di precedenza minore o uguale all'indice in primo argomento: dopo $-$, se c'è un operatore, fy avrà precedenza 200 o più piccola; se c'è $+$ la stessa cosa.
    - Posso scrivere $--4$ oppure $++2$ perché: il primo $-$ ha priorità 200 e quindi può succedere un qualcosa che ha priorità uguale o inferiore; succede un altro $-$; succede un numero 4, con priorità pari a zero. Il fatto che si possa scrivere non vuole dire che sia giusto o sbagliato, ma la grammatica lo permette.
    - Un caso fx (non nell'esempio ma lo vediamo lo stesso), vorrebbe a dire che un qualcosa che succede deve essere strettamente minore dell'ordine di precedenza corrente: non potremmo scrivere $--a$ o $--3$, siccome il secondo $-$ ha anche lui precedenza 200; errore sintattico.
- $(, yf, +), (, xf, -)$
    - yf, xf descrivono operatori postfissi.
- $(, yf, !)$
    - Un simbolo che indica il fattoriale $!$ postfisso, può avere una precedenza che scegliamo noi, in base a quello che più ci garba.
- $(700, xfx, <), (700, xfx, \leq), (700, xfx, =), (700, xfx, \geq), (700, xfx, >)$
  $(500, yfx, +), (500, yfx, -), (400, yfx, *), (400, yfx, /)$
    - Sono tutti operatori comuni infissi.

02/05/2023