Laboratorio di Intelligent Systems: Prolog

Davide Lanti

October 18, 2017

Esercizio 1

Considera l'insieme di symboli di funzione $\mathcal{F} = \{0/0, s/1\}$. Definiamo l'*insieme* $\mathcal{T}(\mathcal{F}, \emptyset)$ dei numeri di Peano come il più piccolo insieme che soddisfi le seguenti proprietà:

- $0 \in \mathcal{T}(\mathcal{F}, \emptyset)$;
- $s(x) \in \mathcal{T}(\mathcal{F}, \emptyset)$, se $x \in \mathcal{T}(\mathcal{F}, \emptyset)$.

Possiamo immaginare una biiezione $P : \mathbb{N} \to \mathcal{T}(\mathcal{F}, \emptyset)$ tra i numeri naturali ed i numeri di Peano:

- $P(0) = 0$;
- $P(n + 1) = s(P(n))$, se $n \in \mathbb{N}$.

Ad esempio, $P(3) = s(s(s(0)))$.

In questo esercizio implementeremo alcuni operatori aritmetici per l'algebra definita sui numeri di Peano.

- Scrivere la definizione di un predicato $sum(X, Y, Z)$ che calcoli la somma Z di due numeri di Peano X ed Y, dove Z è definita come il numero di Peano per cui $P^{-1}(X) + P^{-1}(Y) = P^{-1}(Z)$;
  - E.g., $sum(s(0), 0, Z)$ è vera per l'assegnamento $Z = s(0)$.
- Scrivere la definizione di un predicato $minus(X, Y, Z)$ che calcoli la sottrazione Z di due numeri di Peano X ed Y, dove Z è definita come il numero di Peano per cui $P^{-1}(Z) = P^{-1}(X) - P^{-1}(Y)$;
- Scrivere la definizione di un predicato $mul(X, Y, Z)$ che calcoli la moltiplicazione Z di due numeri di Peano X ed Y, dove Z è definita come il numero di Peano per cui $P^{-1}(Z) = P^{-1}(X) * P^{-1}(Y)$.

Esercizio 2

Il termine Prolog $[a_1, \ldots, a_n]$ denota una lista $(a_1, \ldots, a_n)$ di n elementi. Il termine Prolog $[head|tail]$ denota una lista $(head) \cdot tail$, dove $\cdot$ è l'operatore di concatenazione tra due liste.

- Implementa un predicato $membro(X, L)$ per verificare l'appartenenza di un elemento X alla lista L;
- Implementa un predicato $append(L1, L2, L)$ che assegni alla variabile L la concatenazione delle liste $L1$ ed $L2$.

I predicati $membro/2$ ed $append/3$ possono essere usati per enumerare tutti gli elementi di una lista, o per generare tutte le possibili concatenazioni tra due liste che generino un determinato risultato.

- Implementa un predicato $enumera(L, X)$ che enumeri tutti gli elementi X di una lista L;
- Implementa un predicato $all\_appends(L1, L2, L)$ che enumeri tutti i possibili assegnamenti per $L1$ ed $L2$ tali per cui la concatenazione degli assegnamenti per $L1$ ed $L2$ abbia come risultato la lista L.

Esercizio 3

I predicati `bagoff`, `findall`, e `setof` permettono di assegnare ad una variabile di output *tutti* i possibili assegnamenti di variabili che soddisfino un determinato atomo goal.

- Leggi le descrizioni dei predicati built-in `bagoff`[1], `findall`[2], e `setof`[3] dalla documentazione di `swi-prolog`;
- Esegui gli estratti di codice relativi a questo esercizio dal file

  `2017-IntSys-Lab2.prolog`;

- Implementa un predicato $all\_appends(L1, L2, L, Results)$ che enumeri tutti i possibili assegnamenti per $L1$ ed $L2$ tali per cui la concatenazione degli assegnamenti per $L1$ ed $L2$ abbia come risultato la lista L, e che salvi questi assegnamenti nella variabile $Results$.

[1] `http://www.swi-prolog.org/pldoc/doc_for?object=bagof/3`
[2] `http://www.swi-prolog.org/pldoc/doc_for?object=findall/3`
[3] `http://www.swi-prolog.org/pldoc/doc_for?object=setof/3`

Esercizio 4

I predicati `assert/1` e `retract/1` permettono di cambiare la definizione della base di conoscenza Prolog a tempo di esecuzione.

- Leggi le descrizioni dei predicati built-in `assert/1`[4] e `retract/1`[5] dalla documentazione di `swi-prolog`;
- Esegui gli estratti di codice relativi a questo esercizio dal file

  `2017-IntSys-Lab2.prolog`

  .

[4] `http://www.swi-prolog.org/pldoc/man?predicate=asserta/1`
[5] `http://www.swi-prolog.org/pldoc/doc_for?object=retract/1`