Programmazione logica con vincoli

La programmazione logica con vincoli (CLP) estende la programmazione logica, introducendo il linguaggio di vincoli. Un linguaggio di vincoli è una quintupla:

$$< D, C, A, V, sat, post, label >$$

dove:

- $D \neq \emptyset$ è il dominio del linguaggio di vincoli;
- $C \neq \emptyset$ è l'insieme dei simboli di vincolo;
- $A \neq \emptyset$ è l'insieme degli atomi;
- $V \neq \emptyset$ è l'insieme delle variabili;
- $sat, post, label$ sono 3 funzioni;

A e V sono utilizzate per costruire i termini.

- La funzione sat è in grado di verificare la soddisfacibilità di un CSP. Essa da 2 possibili soluzioni:
  - $\top$ (top) quando è soddisfacibile o potrebbe esserlo.
  - $\bot$ (bottom) quando sicuramente non è soddisfacibile.
- la funzione $post$ serve per manipolare i CSP (aggiungendo dei vincoli) che è possibile costruire con il linguaggio di vincoli a disposizione.
- la funzione $label$ ha un comportamento simile a $post$, ma è non deterministica. Fare $label$ è l'unico modo per capire se un CPS è risolubile

Il linguaggio di vincoli molto usato è FD (Finite Domain), esso è un linguaggio dove:

- I simboli: `#=`, `#\=`, `#<`, `#=<` e altri permettono di imporre vincoli di confronto tra numeri interi.
- Il simbolo `in` permette di attribuire un dominio ad una variabile.

- Il simbolo `ins` permette di attribuire un dominio ad un insieme di variabili elencate in una lista.
- Il simbolo `all_distinct` permette di imporre che la lista sia formata da variabili tutte diverse.

Per usare questa estensione è necessario includerla con: `:- use_module(library(clpfd)).`

esempio: uso di CLP(FD) per scrivere un predicato che sia soddisfatto da una lista e la relativa lunghezza.

```
length([], 0).
length([_,L], length) :-
  length #>= 0,
  N1 #= length - 1,
  length(L, N1).
```

Un'altro linguaggio di vincoli molto interessante è Dif. Esso permette di imporre che il termine a primo argomento sia sempre diverso dal termine al secondo argomento.

CLP(Dif) permette spesso di ovviare ai noti problemi di negazione dei vincoli.

esempio:

```
nevermember(_, []).
nevermember(X, [H|R]) :-
  dif(X, H),
  nevermember(X, R).
```