Introduzione

Programmazione

- strutturata
  - procedure, funzioni, metodi
  - si specifica come ricavare le informazioni
- dichiarativa
  - si definiscono
    - **fatti**: sentenze presupposte sempre vere
    - **regole**
  - la macchina trova da sè la soluzione attraverso le query
  - **knowledge base**: fatti + regole

PROLOG

- simile a SQL
  - inner vs outer join
- richiesta di informazioni tramite query
- non specifichiamo come la macchina deve prendere le informazioni
- versione usata: `SWI_PROLOG`
- estensione `.pl` -> file di knowledge base
- I fatti vengono verificati tramite la sensoristica, le regole vengono definite dagli esperti del dominio del problema.

Esempio di codice

```
# Fatti sempre veri (comandi terminati con un '.')
sing_a_song(anna).          # Anna sta cantando
listens_to_music(sandro).   # Sandro ascolta la musica

# Regole (':-' è un if)
# Anna sing_a_song -> anna listen_to_music
listen_to_music(anna) :- sing_a_song(anna).
# 'anna' -> termine, 'happy'
happy(anna) :- sing_a_song(anna).
happy(sandro) :- listens_to_music(sandro).
# Se Sandro ascolta la musica -> va a suonare
playes_guitar(sandro) :- listens_to_music(sandro).
```

Variabili

- ogni frase che inizia con lettera maiuscola viene interpretata come variabile
  - racchiudere tra apici per ovviare
    - `Anna` -> variabile
    - `'Anna'` -> termine, costante

- una parola non può iniziare con un `_`
    - definisce variabili anonime (non vi si può assegnare valore)

Interrogazioni

```
sing_a_song(anna).      # True -> verificato dal fatto
happy(anna).            # True -> verificata dalle regole

# 'Chi' è una variabile, quindi prolog tenterà di associare sequenzialmente dei termini ad essa scansionando il knowledge base in modo top-down
# con ';' si cerca il successivo, 'Enter' termina il comando
happy(Chi).

# In questo caso 'chi' non è una variabile
happy(chi).         # False

# Essendo la variabile anonima, prolog ci dice che c'è almeno un valore true, ma non può mostrarcelo
# Si può cmq iterare il knowledge base con ';'
happy(_).           # True
```

Fatti

- sentenze che assumiamo come vere
- sintassi: `relation(object1,object2,...)`
    - `cat(tom)`
    - `loves_to_eat(roberto,pasta)` -> relazione, chi, cosa (roberto loves_to_eat pasta)
    - `smart(elisa).` -> elisa è intelligente
    - `of_color(hair,black).`

Regole

- esprimono una relazione implicita (= implicante) tra gli oggetti
- se la condizione a destra è vera -> lo è anche quella a sinistra
    - esempi
        - Lil is happy if she dances.
        - Tom is hungry if he is searching for food
- implicazione (if, è implicato da) -> ':-'
- congiunzione (and) -> ','
    - tutte e due le parti devono essere vere
    - `P :- Q,R` -> P è vero se Q e R è vero
- disgiunzione (or) -> ';'
    - può essere vera una sola delle due parti
    - `P :- Q;R` -> P è vero se Q o R è vero

Knowledge base

- collezione di fatti e regole

```
# Caricare un Kb (da swipl)
[NAME].
```