Linguaggi di Programmazione

Linguaggi

- **LINGUAGGI IMPERATIVI**
- **LINGUAGGI LOGICI**
- **LINGUAGGI FUNZIONALI**

Ogni categoria può contenere linguaggi Object-Oriented

Linguaggio	***Base***	***Concetto di Variabile***	***Stile***	***Programma***	***Esempi***
IMPERATIVO	*Dichiarazione → Istruzioni*	*Astrazione cella memoria fisica*	*Prescrittivo*	*Algoritmi + Strutture Dati*	*C*
LOGICO	*Deduzione logica*	*Matematico*	*Dichiarativo*	*Conoscenza + Controllo*	*PROLOG*
FUNZIONALE	*Applicazione di funzioni*	*Matematico*	*Dichiarativo*	*Comp. Funzioni + Ricorsione*	*LISP*

- **Stile Prescrittivo**
  - **Prescrive** le operazioni che il processore deve eseguire per modificare lo stato del sistema (es. assegnamento)
  - **Esegue** le istruzioni nell'ordine in cui appaiono nel programma (eccezione: strutture di controllo)
- **Stile Dichiarativo**
  - La conoscenza del problema è espressa indipendentemente dal suo utilizzo
  - Alta modularità e flessibilità

Aspetti comuni linguaggi Logici e Funzionali

- Manipolazione **simbolica, non numerica**
- Basati su **concetti matematici**
- **Stile dichiarativo**

Ambienti Run-time

- Per eseguire un programma il S.O. deve mettere a disposizione un **ambiente run-time**
  - Mantenimento dello stato della computazione
  - Gestione della memoria fisica e virtuale
    - **Stack** → **Gestione chiamate** (soprattutto **ricorsive**) a **procedure**
    - **Heap** → **Gestione strutture dinamiche**

Stack e Valutazioni di procedure

- **ACTIVATION FRAMES**
  - **Associazione** di valori ai parametri formali (**Assenza di effetti collaterali** → **Passaggio dei parametri SOLO per VALORE**)
  - **Valutazione ricorsiva** del corpo (tenendo presente che i legami **statici** delle variabili con i loro valori)
  - **Restituzione del/i valore/i del corpo** → Quando il valore vieni ritornato lo Stack subisce una "pop" e l'Activation Frame viene **rimosso**

Return address
Registri . .
Static link
Dynamic link
Argomenti . .
Variabili/definizioni locali, valori di ritorno . .

Regole d’inferenza e Calcoli Logici

Introduzione

- **Calcolo Logico** → Insieme di regole d’inferenza
  - Manipolare formule logiche in modo **sintattico** → Stabilire connessione tra insieme di **formule di partenza (assiomi)** e insieme di **conclusioni**
  - Permette di generare **espressioni sintattiche** a partire da degli assiomi
  - Manipola i seguenti elementi →
    - **Sintassi**
      - Un insieme di proposizioni P
      - Un insieme di formule ben formate FBF, tale che P $\subseteq$ FBF
      - Un sottoinsieme di assiomi A $\subseteq$ FBF
      - Un insieme di regole di inferenza che ci permettono di incrementare FBF
    - **Semantica**
      - Una funzione di verità che ci permette di distinguere ciò che è vero da ciò che è falso rispetto a una data interpretazione
      - Funzione di Interpretazione: V (o I)
      - Tavole di verità
- **2 tipi di logica** →
  - **Logica Proposizionale**
  - **Logica dei Predicati del Primo Ordine**

Logica Proposizionale

- La **logica proposizionale** si occupa delle **conclusioni** che possiamo trarre da delle **proposizioni**
- Una **logica proposizionale** è **sintatticamente** definita da un insieme **P** di proposizioni
  - Esempi:
    - **P** = {AB = BC, ∠ABH = ∠HBC, BH = BH}
    - **P** = {piove, l’unicorno è un animale mitico}
    - **P** = {p, q, r, s, w}
- All’insieme **P** è associata una funzione di verità, o di valutazione, ***V***

  $V : P \rightarrow \{vero, falso\}$

  che **associa un valore di verità ad ogni elemento di** **P** (cioè ad ogni proposizione)
- La funzione di **valutazione** è il ponte di connessione tra la **sintassi** e la **semantica** di un linguaggio logico
  - Esempi
    - *V*(q) = *vero*, *V*(p) = *vero*, *V*(w) = *falso*
    - *V*(l’unicorno è un animale mitico) = *vero*
    - *V*(piove) = *falso*
- Connettivi logici
  - Congiunzione
  - Disgiunzione
  - Negazione
  - Implicazione

- **Formule ben Formate (FBF)** → Insieme di tutte le formule formate dagli elementi di P e dalle loro combinazioni
- **Letterali** → Formule atomiche/Negazioni dell'insieme P
- **Parte Semantica** → Funzione di verità *(dice ciò che è vero e falso di un insieme di proposizioni)*

Regole d'inferenza

- Una regola di inferenza ha la seguente forma generale:

FBF

$$\frac{F_1, F_2, \dots, F_k}{R} \quad \text{[nome regola]}$$

Formula generata

Regola d'inferenza

- **Modus Ponens**

$$\frac{p \Rightarrow q, \quad p}{q} \quad \text{[modus ponens]}$$

  - *p ⟹ q:* **Se** piove, **allora** la strada è bagnata
  - p: piove
  - *q*: (allora) la strada è bagnata

- **Modus Tollens**

$$\frac{p \Rightarrow q, \quad \neg q}{\neg p} \quad \text{[modus tollens]}$$

  - *p ⟹ q*: **Se** piove, **allora** la strada è bagnata
  - ¬*q*: la strada non è bagnata
  - ¬*p*: (allora) non piove

- Queste regole d'inferenza fanno parte del **CALCOLO NATURALE** → *Permettono di derivare "direttamente" una formula ben formata mediante una sequenza di passi ben codificati*

Principio di Risoluzione

- Opera su FBF in **forma normale congiunta**
- Ognuno dei congiunti viene chiamato **clausola**

$$\frac{p \vee \neg r, \quad s \vee r}{p \vee s}$$

Clausola *risolvente*

$$\frac{\neg r, \quad r}{\perp}$$

Clausola *vuota*

- **Principio di Risoluzione unitario** → Si ha quando una delle due clausole è un letterale
- **Esempio**
    - (Da) <*Non piove*>, <*piove o c'è il sole*>
      (Segue che) <*C'è il sole*>

$$\frac{\neg p, \quad q_1 \vee q_2 \vee \cdots q_k \vee p}{q_1 \vee q_2 \vee \cdots q_k} \quad \text{[unit resolution]}$$

$$\frac{p, \quad q_1 \vee q_2 \vee \cdots q_k \vee \neg p}{q_1 \vee q_2 \vee \cdots q_k} \quad \text{[unit resolution]}$$

Dimostrazioni per assurdo

- Supponiamo di avere a disposizione un insieme di formule FBF (vere, data una certa interpretazione V)
- Supponiamo di voler dimostrare che una certa proposizione p (o formula atomica) è vera
- Possiamo procedere usando il metodo della dimostrazione per assurdo
    - **Assumiamo che ¬p sia vera**
    - **Se**, combinandola con le proposizioni in FBF **ottengo una contraddizione**, allora concludo che **p deve essere vera**

Assiomi

- Alcune proposizioni sono sempre vere, indipendentemente dalla loro interpretazione (tautologie)
    - **A1**: $A \Rightarrow (B \Rightarrow A)$
    - **A2**: $(A \Rightarrow (B \Rightarrow C)) \Rightarrow ((A \Rightarrow B) \Rightarrow (A \Rightarrow C))$
    - **A3**: $(\neg B \Rightarrow \neg A) \Rightarrow ((\neg B \Rightarrow A) \Rightarrow B)$
- Alcune tautologie sono codificabili come regole di inferenza e viceversa
    - **A4:** $\neg(A \wedge \neg A)$ principio di non-contraddizione
    - **A5:** $A \vee \neg A$ principio del terzo escluso

Sintassi e Semantica

- Un **calcolo logico** fornisce una manipolazione **sintattica**
    - **DERIVAZIONE** $\vdash$
- Una **funzione di valutazione** caratterizza la **semantica**
    - **CONSEGUENZA LOGICA (ENTAILMENT)** $\models$
- **Teorema di Completezza e Validità** → **S $\vdash$ *f* se e solo se S $\models$ *f***

- ("**S va in f se e solo se f è conseguenza logica di S"),** dove **S** è un insieme di formule iniziale ed ***f*** è una FBF; Il tutto in dipendenza da una particolare funzione di verità ***V***

Tautologie e Modelli

- Particolare **interpretazione *V*** che rende vere tutte le formule in **S** → **Modello** di **S**
- **FBF sempre vera indipendentemente dal valore assegnato dei letterali** → **Tautologia**

NB: *Una tautologia è vera «in» ogni modello*

Logica del Primo Ordine

- Un **linguaggio logico del primo ordine** è costituito da **termini** costruiti a partire da:
  - **V** → Insieme di simboli di Variabili
  - **C** → Insieme di simboli di Costante
  - **R** → Insieme di simboli di Relazione o Predicati
  - **F** → Insieme di simboli di Funzione
  - **Connettivi Logici**
- La costruzione di un **linguaggio logico del primo ordine è RICORSIVA**
- **Predicati** → $r \subseteq C_0 \times C_1 \times \ldots \times C_k$ → *ovvero relazioni cartesiane su C, scritte come* $r(c_1, c_2, \ldots, c_k)$
- Le funzioni sono definite con il seguente dominio e codominio

  $f : \mathbf{C}_0 \times \mathbf{C}_1 \times \ldots \times \mathbf{C}_m \rightarrow \mathbf{C}$

  una funzione si scrive come $f(c_1, c_2, \ldots, c_m)$ → **UNA FUNZIONE NON È UNA FBF**
- **FBF** →
  - Un **termine** t_j può essere un elemento di **C**, di **V**, oppure un'applicazione di funzione $f(t_1, t_2, \ldots, t_s)$
  - Un termine costituito da un **predicato** $r(t_1, t_2, \ldots, t_k)$, dove ogni t_i è un termine, appartiene ad **FBF**
  - Diversi elementi di **FBF** connessi dai connettivi logici standard (congiunzione, disgiunzione, negazione, implicazione) appartengono ad **FBF**
  - Denotiamo con $\boldsymbol{t}(t_1, t_2, \ldots, t_r)$ tale combinazione di termini
  - Le formule

    $\forall x . \boldsymbol{t}(t_1, t_2, \ldots, x, \ldots, t_r)$ e $\exists x . \boldsymbol{t}(t_1, t_2, \ldots, x, \ldots, t_r)$ appartengono ad **FBF**

Prolog

Introduzione

- **Base formale:**
  - Calcolo dei predicati del primo ordine → (Solo **clausole di Horn**)
  - Tecniche per risoluzione teoremi
- Ogni FBF può essere riscritta in **Forma Normale a Clausole:**
  - ***Forma Normale Congiunta*** → Congiunzione di disgiunzioni
  - ***Forma Normale Disgiunta*** → Disgiunzione di congiunzioni

$$\underbrace{(p(x) \lor q(x,y) \lor \neg t(z))}_{\text{clausola 1}} \land \underbrace{(p(w) \lor \neg s(u) \lor \neg r(v))}_{\text{clausola 2}} \quad \longrightarrow \quad p(x) \lor q(x,y) \lor \neg t(z)$$

$$\underbrace{(\neg t(z))}_{\text{clausola 1}} \land \underbrace{(p(w) \lor \neg s(u))}_{\text{clausola 2}} \land \underbrace{(p(x) \lor s(x) \lor q(y))}_{\text{clausola 3}} \quad \longrightarrow \quad p(w) \lor \neg s(u) \lor \neg r(v)$$

- Oppure possono essere riscritte come congiunzione di implicazioni:

$$t(z) \Rightarrow p(x) \lor q(x,y)$$
$$s(u) \land r(v) \Rightarrow p(w)$$

- **Clausole di Horn** → *Insieme di disgiunzioni di letterali che hanno al più un letterale positivo (con o senza letterali negativi)*
- Programma Prolog:
  - Insieme di **fatti e regole**
  - Fornisce informazioni su un sistema → **Knowledge base**
  - **Non si esegue, ma si interroga → Queried**
- **Sintassi**: un programma Prolog è costituito da un insieme di clausole della forma

```
a.                         % FATTO o ASSERZIONE
c :- b1, b2, …, bn.        % REGOLA
:- q1, q2, …, qm.          % GOAL
?- q1, q2, …, qm.          % QUERY
```

  - In cui **a**, **b**$_i$, **c**, e **q**$_i$ sono **termini** (composti)
  - Il prompt Prolog è anche un operatore che chiede al sistema di valutare il **goal**, in questo caso una congiunzione di termini
- **Termini:**
  - **Atomi**
  - **Variabili**
  - **Composizione di termini → Termine composto** (simbolo di **funtore** + uno o più argomenti)