

Informatica divisa in due mondi:

- Teoria del calcolo (simbolico)

- Nasce dalla logica

- Modello algebrico  $\mathcal{L} \rightarrow \mathcal{S}$

- Teoria delle macchine calcolatrici

- Macchine (fisiche e non) per eseguire calcoli

- Basate su macchina di  $\forall \mathcal{N}$

- Load-state model

- Circuiti logici per calcolo

} memoria



Universalità

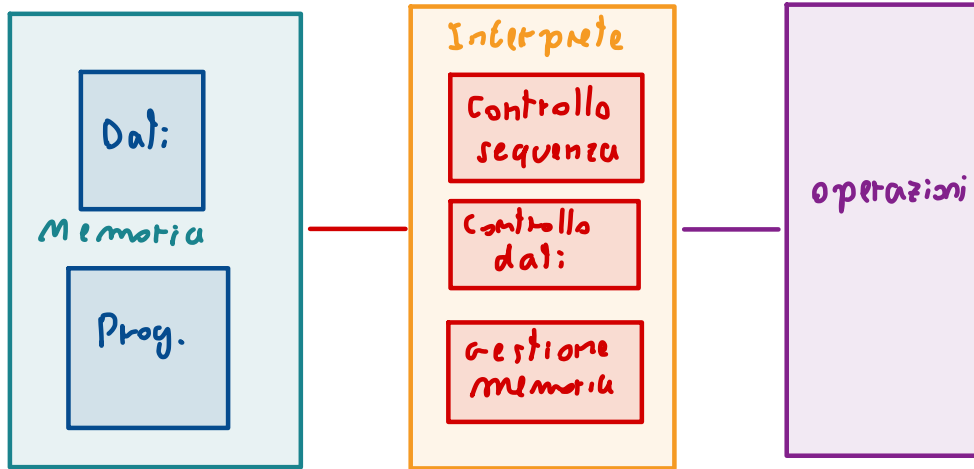
Linguaggio di: programmazione

- sintassi
- semantica
- pragmatica
- 

· linguaggio comprensibile  
da macchina calcolatrice

Domanda. È la computazione simbolica eseguibile su  
una macchina?

Def. Una **macchina astratta** per un linguaggio  $L$  è un insieme di: strutture dati; e algoritmi per eseguire programmi scritti in  $L$



Una macchina  $M$  è una macchina per un linguaggio  $L$ ,  
il suo **linguaggio macchina**

**Implementare**  $L = \text{date } M_L$  (macchina per  $L$ )

Es. Macchina fisica  $\mathcal{H}$

La realizzazione di  $M$  può poi avvenire a differenti livelli

- Realizzazione in hardware (costruzione fisica)
- Simulazione

Osservazioni Più espressivo è  $L$ , più complessa sarà  $M_L$

- La macchina determina la nozione di **esecuzione** di un programma
- La macchina è **imperativa**  $\Rightarrow$  possiamo definire il concetto di esecuzione tramite il concetto di **stato della macchina**

Def. Lo **stato** di una macchina  $M$  è una rappresentazione astratta di tutta l'informazione di  $M$  in un certo istante necessaria all'esecuzione di un programma

$\rightarrow$  Più espressivo è  $M_L$ , più complessa sarà la nozione di stato

Ex. Stato = memoria + indirizzo PC + ...

Come descrivere l'esecuzione di programmi in  $\mathcal{L}$ ?

① Date  $M_{\mathcal{L}} \rightsquigarrow$  ambigua (quali aspetti di  $M_{\mathcal{L}}$  sono rilevanti?  
tempo esecuzione? spazio? ...)  
non modulare, ...

② Identificare una nozione di **stato** sufficientemente ricca  
da descrivere l'esecuzione di programmi in  $\mathcal{L}$

$$\langle \sigma, P \rangle \rightarrow \langle \sigma', P' \rangle$$

structural operational semantics (SOS)

NB. La descrizione della SOS dovrebbe essere sufficientemente precisa  
da specificare l'implementazione dell'interprete di  $\mathcal{L}$  in  
una qualsiasi macchina  $M$  con stato adeguato

## Linguaggio di Programmazione

- Sintassi
- Semantica
- 

↳ riferimento alla macchina

→ model-dependent

(equivalenza estensionale tra macchine) ✓

In generale

sintassi → grammatiche, ...

semantica → **trasformazione stato**

Per capire un linguaggio naturale, sintassi non è abbastanza

- nomi
- verbi
- aggettivi
-

Un linguaggio di programmazione è un meccanismo di astrazione della macchina fisica

Meccanismo di base: nome

La macchina astratta consiste di vari "oggetti" (oggetti ( $M$ ))

- variabili
  - indirizzi; memoria
  - operatori
  - strutture controllo
  - ⋮
- } natura imperativa

Def. Un nome è una espressione in  $L$  (solitamente un identificatore) per rappresentare un oggetto

NB. L'oggetto non è necessariamente un oggetto di  $M$



F<sub>1</sub> Se  $M_L$  ha una operazione per la somma, l'identificatore +  
in  $L$  è un nome per questa

Es. - Preso un linguaggio imperativo  $L$ ,

int  $x$

dice:

$x$  è un identificatore per una variabile (oggetto di:  $M_L$ )  
(nome)

Bisognerebbe parlare di: variabile di nome  $x$

- in OCaml

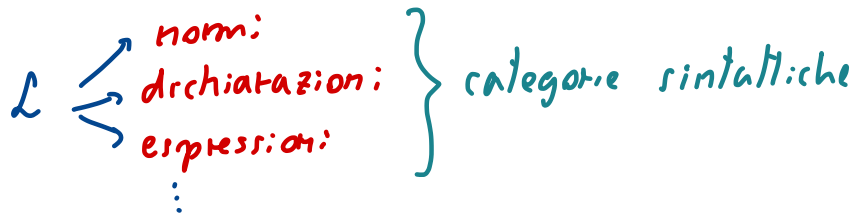
def  $f = \text{fun } y \rightarrow y$  } la funzione di nome  $f$

$\implies$  Abbiamo oggetti quali: variabili e funzioni

I linguaggi con nomi offrono meccanismi di astrazione tramite nomi.  
Chiamiamo questi meccanismi dichiarazioni (o definizioni)

Es. In Ocaml

let <identificatore> = <exp> è una dichiarazione



Domanda A quali oggetti un linguaggio può dare nome?

Def. Gli oggetti a cui un linguaggio può dare nome sono detti oggetti denotabili

Domanda. Dato un linguaggio  $L$  con dichiarazioni, cosa serve  
(allo stato della) macchina astratta per eseguire dichiarazioni?

Def. L'ambiente è l'insieme delle associazioni:  
(norme, oggetto denotabile)

esistente a run-time in uno specifico punto del programma

NB. L'ambiente non esiste nella macchina fisica. È una caratteristica  
del linguaggio di alto livello che deve essere simulato  
dall'implementazione

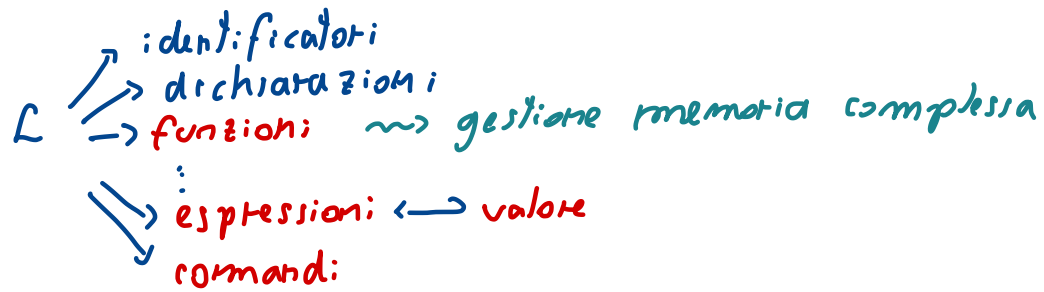
Se  $\mathcal{L}$  ha dichiarazioni, verosimilmente la semantica operativa usa la nozione di ambiente

$$\langle \eta, P \rangle \rightarrow \langle \eta', P' \rangle$$

$M_{\mathcal{L}}$  deve quindi gestire **ambiente** e **dichiarazioni**:

- blocchi  $( \{e\} \ x = \langle \dots \rangle \text{ in } \langle \dots \rangle )$
- scope  $( \{e\} \ x = \langle \dots \rangle \text{ in } \langle \dots \rangle )$

$\rightsquigarrow$  Gestione di queste cose in  $M_{\mathcal{L}}$  non facile.



Def. Una **espressione** è una entità sintattica la cui **valutazione** produce un **valore** oppure **non termina**

v.B. Def. relativa alla semantica su macchina astratta

Def. Un **comando** è una entità sintattica la cui **valutazione** non necessariamente restituisce un valore, ma può avere un **effetto collaterale**

Ma cos'è allora un effetto collaterale?

$$\langle (\sigma_1 \dots \sigma_m), P \rangle \rightarrow \langle (\sigma_1' \dots \sigma_n'), P' \rangle$$

componenti  
dello stato  
della macchina

Un effetto collaterale è relativo a una **porzione di stato** (e.g. memoria)

Diciamo che  $P$  ha effetto collaterale se

$$\langle (\sigma, \varepsilon), P \rangle \rightarrow \langle (\sigma', \varepsilon'), P' \rangle$$

e  $\varepsilon \neq \varepsilon'$  (modifica stato)

Domanda. Quando è che problematico avere effetti collaterali?

Risposta. Quando le parti interessate dello stato sono oggetti denotabili

Def. Una variabile è un riferimento (e.g. indirizzato) che può contenere valori

NB. Le variabili sono immutabili

Nei linguaggi imperativi le variabili sono oggetti denotabili

L'assegnamento è un comando con effetti collaterali sulla porzione di stato della memoria che associa valori memorizzabili a locazioni