Notazione:
$Prog^L$ = Insieme di programmi scritti in L
D = Insieme dei dati in input e output
$P^L \in Prog^L$, PL: D → D parziale ricorsiva tale che P^L(Input) = Output

1. Definire cosa è un interprete: Dare definizione semantica e descrivere la struttura

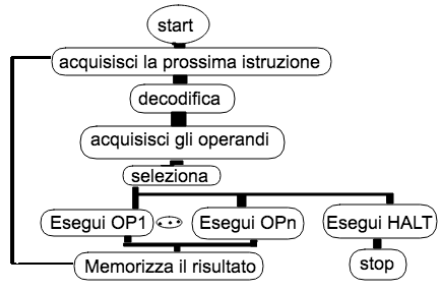
Un interprete per un linguaggio L, scritto nel linguaggio Lo è un programma che realizza:
$I^{Lo,L}$: ($Prog^L$ x D) → D tale che

$$I^{Lo,L}(P^L, Input) = P^L(Input)$$

Definizione semantica:

Un programma int ∈ L tale che, per ogni S-programma P e per ogni dato $d \in D$: $[\![int]\!]^L(P,d) = [\![P]\!]^S(d)$ è un interprete in L di S-programmi (o semplicemente di S).

Struttura:



Si realizza l'interprete di M_L mediante un insieme di istruzioni in Lo, ovvero si realizza un programma $I^{Lo,L}$, scritto in Lo ed interpreta istruzioni di L.

Definire intuitivamente e formalmente il concetto di specializzazione e specializzatore

Nella specializzazione si effettua un primo passaggio chiamato *Valutazione Parizale,* consiste nell'attuare trasformazioni all'interno dello stesso linguaggio per migliorarne l'efficienza.
Un programma P(X,Y) di cui X noto, possiamo trasformarlo in PX(Y) dove le computazioni relative a X sono state svolte prima dell'esecuzione.

Specializzatore:
Uno specializzatore per programmi in un linguaggio L è un programma che realizza la funzione:
$Spec_L$: ($Prog^L$ x D) → $Prog^L$ tale che, $P^L \in Prog^L$, d∈D

$$Spec_L(P^L,d) = P_d \text{ e } I_L(P^L(d,Y) = I_L(P_d,Y)$$

Dove Y∈D, I_L è l'interprete per L.

Definizione semantica:
Un programma *spec* scritto in un linguaggio di implementazione L è uno specializzatore *spec* ∈ L se, per ogni programma P scritto in un linguaggio di programmazione S e per ogni porzione di suo input s ∈ D, restituisce un T-programma: $[\![spec]\!]^L(P,s)$ tale che per ogni dato d ∈ D:

$$[\![[\![spec]\!]^L(P,s)]\!]^T(d) = [\![P]\!]^S(s,d)$$

Cosa significa implementare un linguaggio L? Definire esplicitamente e dettagliatamente tutti i concetti utilizzati

Implementare un linguaggio L significa realizzare una macchina astratta M_L per L. M_L è un dispositivo che permette di eseguire programmi scritti in linguaggio L.
Una macchina astratta corrisponde univocamente ad un linguaggio, il suo linguaggio macchina.
Dato un linguaggio L esistono infinite macchine astratte che hanno come linguaggio macchina L (differiscono nel modo in cui l'interprete viene realizzato e nelle strutture dati utilizzate).

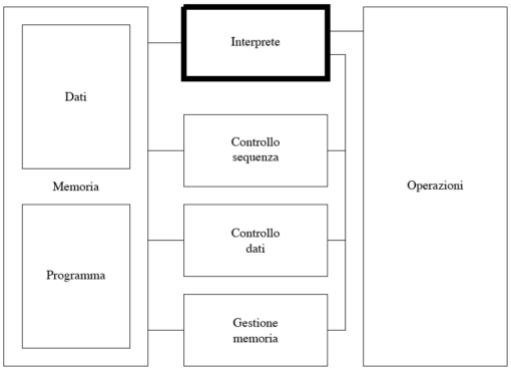
Esistono due principali soluzioni per implementare un linguaggio:

Traduzione implicita (soluzione interpretativa)*:* Realizzata da una simulazione di M_L mediante programmi in Lo.
Usata per piccoli programmi o quando l'efficienza non è importante.

Traduzione esplicita (soluzione compilativa): Realizzata traducendo i programmi in L direttamente in programmi in Lo.
Usata per grandi applicazioni commerciali.

Definire cosa è una macchina astratta: Dare la struttura e la definizione di linguaggio macchina

Dato un linguaggio di programmazione L, la macchina astratta di L (detta M_L), è un insieme di strutture dati ed algoritmi che permettono di memorizzare ed eseguire programmi di L.



Struttura:
M_L = Memoria (immagazzina dati e programmi) + Interprete (esegue le istruzioni dei programmi)

Linguaggio Macchina:
Data una macchina astratta M, il linguaggio L_M che ha come stringhe legali tutte le stringhe interpretabili da M è detto linguaggio macchina.

3. Descrivere intuitivamente cosa è un comando

I comandi rappresentano funzioni da memoria a memoria e devono essere eseguiti per poter attuare la corrispondente trasformazione (irreversibile) della memoria. Due comandi sono equivalenti se, per ogni stato (memoria) in input, producono lo stesso output (memoria) in output. Sono una categoria sintattica i cui elementi sono eseguiti per generare trasformazioni irreversibili di stato.

… ed in particolare un comando iterativo. Definire formalmente i comandi di IMP e dare semantica operazionale statica e dinamica al comando iterativo while

Un comando iterativo permette l'esecuzione ripetuta di un comando o un comando composto mediante iterazione o ricorsione, questi sono due meccanismi che permettono di ottenere formalismi di calcolo turing-completi, senza di essi avremmo automi a stati finiti.
L'iterazione può essere:

- *Indeterminata:* cicli controllati logicamente (while, repeat, …)
- *Determinata:* cicli controllati numericamente, con un numero di ripetizioni del ciclo determinato al momento dell'inizio del primo ciclo.

Comandi di IMP

$$C^V: < Com > \quad C \rightarrow skip \mid I := e \mid C; C \mid if\ B\ then\ C\ else\ C \mid while\ B\ do\ C \mid \{D; C\} \mid P(ae)$$

Semantica statica (while)

$$C_{S3}: \frac{\Delta \vdash_V e: bool\ ,\ \Delta \vdash_V c}{\Delta \vdash_V while\ e\ do\ c}$$

Semantica dinamica (while)

$$C_6: \frac{\rho \vdash_\Delta < e, \sigma > \rightarrow_e^* < true, \sigma >}{\rho \vdash_\Delta < while\ e\ do\ c, \sigma > \rightarrow_c < c; while\ e\ do\ c, \sigma >}$$

$$C_7: \frac{\rho \vdash_\Delta < e, \sigma > \rightarrow_e^* < false, \sigma >}{\rho \vdash_\Delta < while\ e\ do\ c, \sigma > \rightarrow_c \sigma}$$

… ed in particolare un comando condizionale. Definire formalmente i comandi di IMP e dare semantica operazionale statica e dinamica al comando condizionale if

Un comando condizionale appartiene ai comandi di selezione ed è uno strumento che consente di scegliere a tempo di esecuzione quale flusso di istruzioni eseguire tra due o più cammini possibili, in base allo stato dinamico del programma, in particolare si basa sul soddisfacimento di condizioni booleane.

Comandi di IMP

. . .

Semantica statica (if)

$$C_{S2}: \frac{\Delta \vdash_V e: bool\ ,\ \Delta \vdash_V c_0,\ \Delta \vdash_V c_1}{\Delta \vdash_V if\ e\ then\ c_0\ else\ c_1}$$

Semantica dinamica (if)

$$C_3: \frac{\rho \vdash_\Delta < e, \sigma > \rightarrow_e < e', \sigma >}{\rho \vdash_\Delta < if\ e\ then\ c_0\ else\ c_1, \sigma > \rightarrow_c < if\ e'\ then\ c_0\ else\ c_1, \sigma >}$$

$$C_4: \rho \vdash_\Delta < if\ true\ then\ c_0\ else\ c_1, \sigma > \rightarrow_c < c_0, \sigma >$$
$$C_5: \rho \vdash_\Delta < if\ false\ then\ c_0\ else\ c_1, \sigma > \rightarrow_c < c_1, \sigma >$$

… ed in particolare un comando condizionale. Definire formalmente i comandi di IMP esteso con l'aggiunta del comando while. Dare semantica operazionale statica e dinamica a tale comando while

. . .

… ed in particolare un comando condizionale. Definire formalmente i comandi di IMP e dare semantica operazionale statica e dinamica ad un comando iterativo a scelta

. . .

Descrivere intuitivamente cosa è una dichiarazione

Le dichiarazioni descrivono legami (bindings) tra identificatori e denotazioni, infatti devono essere elaborate per attuare la creazione/trasformazione (reversibile) di legami.
Esse denotano/descrivono richieste di modifica/creazione di ambienti e si dicono equivalenti se producono lo stesso ambiente, e la stessa memoria in caso di side-effects, in tutti gli stati di computazione.

… e descrivere i vari tipi di composizione di dichiarazioni in IMP. Definire formalmente le dichiarazioni di IMP e dare semantica operazionale statica e dinamica di una composizione di dichiarazioni a scelta.

Dichiarazioni

$$D^V : < Dec > D \ \rightarrow nil \mid const\ I{:}\tau = e \mid var\ I{:}\tau = e$$
$$\mid D\ in\ D \mid D; D \mid P \mid Proc\ P(form)\ C \mid form = ae$$

(vedi sotto)

... e definire formalmente le dichiarazioni di IMP e dare semantica operazionale statica e dinamica alla dichiarazione di procedura.

4/5. Passaggio per valore

Il valore dell'attuale non è usato per inizializzare il corrispondente parametro formale, che si comporta come una variabile locale. Le modifiche al parametro formale non passano al parametro attuale, viene implementato mediante una copia.

… Passaggio per valore-risultato

A volte chiamato passaggio per copia, i parametri formali hanno uno storage locale, le modifiche vanno poi riportate ai parametri attuali.

… Passaggio per riferimento

Viene passato l'indirizzo di memoria al parametro formale, in pratica il parametro formale è un alias dell'attuale, quindi le modifiche ai parametri sono bidirezionali.

6. Definire il concetto di ricorsione. Si consideri il seguente programma (ecc..)

Una funzione (procedura) è ricorsiva se è definita in termini di se stessa. Ogni programma ricorsivo (iterativo) può essere tradotto in uno equivalente iterativo (ricorsivo).
Una funzione si dice tail recursive se una funzione restituisce il valore senza ulteriore elaborazione.