Lo stato

- I domini della semantica restano invariati: Ide, Val (Int $\cup$ $\{\varpi,\perp\}$) Loc (locazioni di memoria), FunctDecl.
- ma definiamo 2 funzioni:
  - ν_{Loc} :Loc$\rightarrow$ Val che trasforma locazioni in valori
  - ε_{Loc} :Val $\rightarrow$ Loc che trasforma valori in locazioni

Puntatori

$$\text{(Amp)}\quad \frac{\sigma(\text{Ide}) = d}{\langle \&\text{Ide}, \langle \sigma, \mu \rangle \rangle \rightarrow \langle \text{in}_{\text{val}}(d), \mu \rangle}$$

$$\text{(Star)}\quad \frac{\langle \text{Exp}, \langle \sigma, \mu \rangle \rangle \rightarrow \langle v, \mu' \rangle}{\langle *\text{Exp}, \langle \sigma, \mu \rangle \rangle \rightarrow \langle \mu'(\text{in}_{\text{Loc}}(v)), \mu_1 \rangle}$$

Programmazione strutturata

- Problemi complessi si risolvono suddividendo il problema in problemi più semplici ciascuno risolto da un programma
  - es: faiCerchio, faiRettangolo e faiTriangolo sono pezzi di programma che assolvono un compito, in questo caso quello di leggere e calcolare per una data fig. geometrica.
- I programmi definiti dentro un programma più generale si chiamano funzioni (o procedure) .

“Sottoprogrammi”

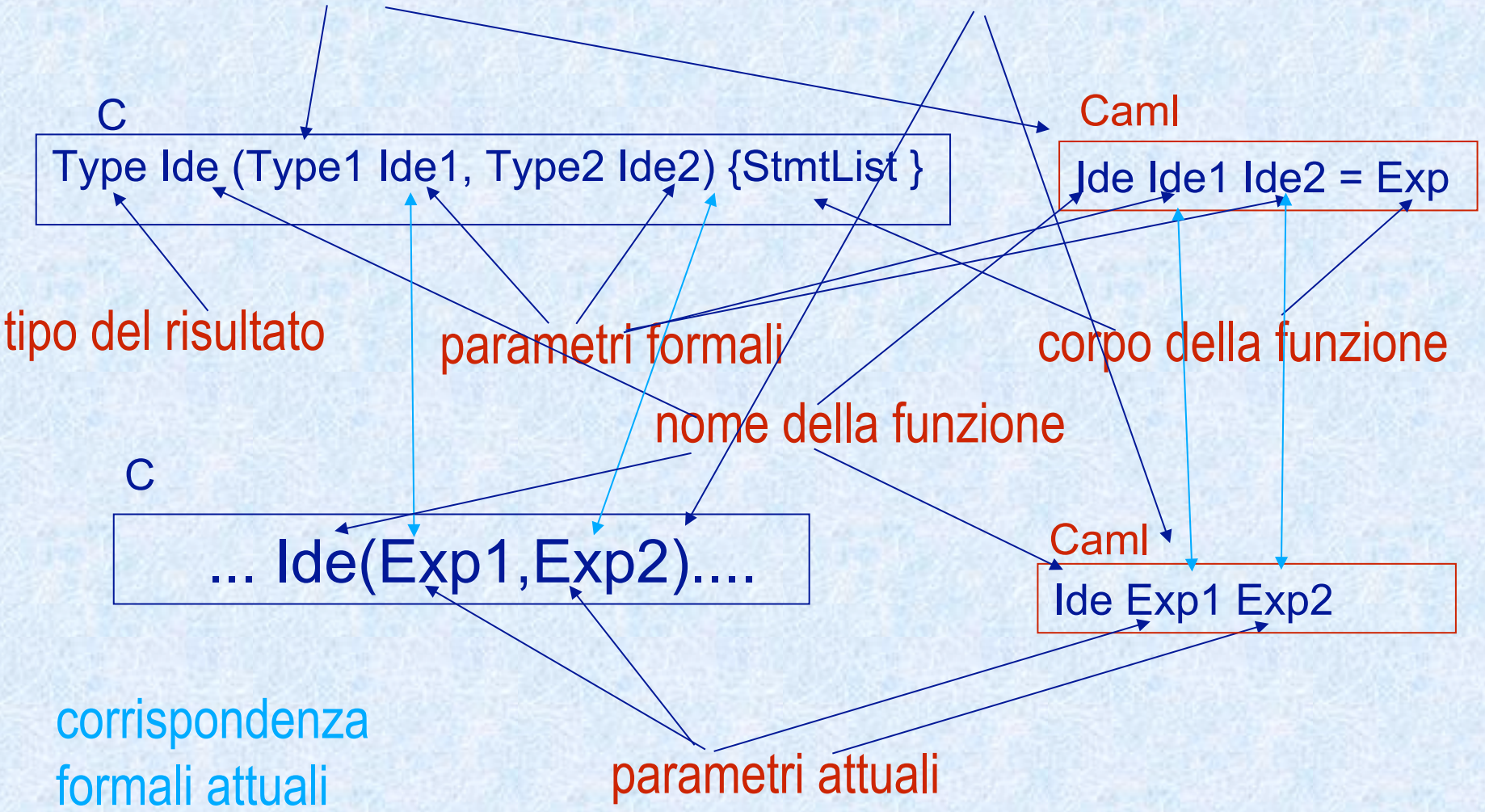
- Trasferimento del controllo (cambia il flusso delle istruzioni)
- Concetti di programma chiamante e programma invocato
- Scambio di dati tra programma invocato e programma chiamante
  - parametri formali e
  - parametri attuali
  - risultato del programma invocato
  - ambiente del programma invocato

Funzioni in ogni linguaggio anche Caml e C

- definizione:
  - nome
  - nome e tipo dei parametri (formali)
  - tipo calcolato
  - corpo
- invocazione:
  - nome
  - i parametri (attuali)

In Caml i tipi non compaiono perchè inferiti dal sistema, il corpo è un'espressione

Definizione e invocazione



Definizioni di funzioni

$$\text{(FunDef)}\ \frac{\sigma' = \sigma\,[\langle x, B, \sigma' \rangle / m],}{\langle \text{Type}_r\ m(\text{Type}\ x)\ B, \langle \sigma, \mu \rangle \rangle \rightarrow \langle \sigma', \mu \rangle}$$

$$\text{(STL)}\ \frac{\langle \text{FunDef}, \langle \sigma, \mu \rangle \rangle \rightarrow \langle \sigma', \mu \rangle \quad \langle \text{StmtList}, \langle \sigma', \mu \rangle \rangle \rightarrow \mu'}{\langle \text{FunDef StmtList}, \langle \sigma, \mu \rangle \rangle \rightarrow \mu'}$$

Espressioni: invocazione di funzione

$$\frac{\langle Exp,\langle \sigma,\mu \rangle\rangle \rightarrow \langle v,\mu' \rangle \qquad \sigma(m)=\langle x,B,\sigma' \rangle \quad free(\mu')=\langle \ell,\mu'' \rangle \quad free(\mu'')=\langle \ell',\mu''' \rangle \quad \langle B, \langle \omega[\ell'/retval][\ell/x].\sigma', \mu'''[v/\ell] \rangle\rangle \rightarrow \mu^{VI}}{\langle m(Exp);, \langle \sigma,\mu \rangle\rangle \rightarrow \langle \mu^{VI}(\ell'), \mu^{VI} \rangle}\ invoc$$

Comandi: return

(return)

$$\frac{}{\langle \text{return Exp}, \langle \sigma, \mu \rangle \rangle \rightarrow \langle \text{retVal=Exp}, \langle \sigma, \mu \rangle \rangle}$$

Scoping:

- Individua l'ambiente di valutazione della funzione:
  - Statico: l'ambiente è costruito a partire dall'ambiente di risultante dalla dichiarazione della funzione
  - dinamico: l'ambiente è costruito a partire dall' ambiente di invocazione della funzione.
- Quasi tutti i linguaggi di programmazione utilizzano lo scoping statico

Passaggio dei parametri:

- La modalità di passaggio dei parametri :
  - per valore : le modifiche effettuate dal chiamato non si ripercuotono nello stato del chiamante. Tutti i linguaggi la prevedono.
  - per riferimento: le modifiche effettuate dal chiamato si ripercuotono nello stato del chiamante. È un modo per calcolare il risultato o parte di esso. Molti linguaggi imperativi la prevedono o la simulano,
  - ne esistono altre (per nome) che non trattiamo.
- In C: passaggio solo per valore. Per riferimento è simulata utilizzando i parametri di tipo puntatore: cioè si passa l'indirizzo. esercizi