PROCEDURE

31/03/14

$C ::= \ldots \mid p(ae)$

$D ::= \ldots \mid \text{procedure } p(form)\,C$

$form ::= \bullet \mid \text{const } x : \tau,\ form \mid \text{var } x : \tau,\ form$

terminatore di passaggio di parametri

$ae ::= \bullet \mid E,\ ae$

NO Procedure ricorsive

$E \in Typ \ni et \qquad et ::= \tau$

$A \in Typ \ni aet \qquad aet ::= \bullet \mid et,\ aet$ → devo associare dei nuovi tipi perché ho dichiarato delle cose nuove

---

$FI_C(p(ae)) = \{p\} \cup FI_{AE}(ae)$

$FI_D(\text{procedure } p(form)\ C) = FI_C(C) \setminus BI_{form}(form)$

$FI_{form}(form) = \emptyset$

$FI_{AE}(\bullet) = \emptyset$

$FI_{AE}(E, ae) = FI_E(E) \cup FI_{AE}(ae)$

SEMANTICA STATICA

$$\frac{form : \Delta_0,\ \Delta[\Delta_0] \vdash c}{\Delta \vdash procedure\ p(form)\ c : [p : T(form)_{proc}]}$$

$\bullet : \phi$

$$\frac{form : \Delta}{const\ x : \tau,\ form : \Delta[x : \tau]}\ x \notin I,\ \Delta : I$$

$$\frac{form : \Delta}{var\ x : \tau,\ form : \Delta[x : \tau_{loc}]}\ x \notin I,\ \Delta : I$$

$T : form \to ATyp$
$T(\bullet) = \bullet$
$T(const\ x : \tau,\ form) = \tau, T(form)$
$T(var\ x : \tau,\ form) = \tau, T(form)$

definisco questa funzione perché nella chiamata della procedura ho tipi ATyp che non possono essere costanti

$$\frac{\Delta \vdash ae : aet}{\Delta \vdash p(ae)},\ \Delta(p) = aet_{proc}$$

vuol dire che ho già dichiarato la procedura prima della chiamata

$\Delta \vdash \bullet : \bullet$

$$\frac{\Delta \vdash \in : \tau \quad , \quad \Delta \vdash ae, aet}{\Delta \vdash \in , ae : \tau , aet}$$

```
[var z : Z = 3; ]D0
[procedure p(var x: bool, var y: int)
        if x then z := y;  ]C1  (C2 = z := y)
             else z := z;       (C3 = z := z)
]D
[p(tt, z+1); ]C0
```

$$\dfrac{\dfrac{\dfrac{\dfrac{\Delta_0[\Delta_1] \vdash z : int \quad \Delta_0[\Delta_1] \vdash 1 : int}{\Delta_0[\Delta_1] \vdash z+1 : int} \quad \dfrac{}{\Delta_0[\Delta_1] \vdash \bullet : \bullet}}{\Delta_0[\Delta_1] \vdash \#, z+1, \bullet : bool, int, \bullet}}{\Delta_0[\Delta_1] \vdash C_0} \quad , \Delta_0[\Delta_1](\rho) = bool, int, \bullet\ proc}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset \vdash 3 : int}}{\emptyset \vdash D_0 : \Delta_0[z : int\ loc] = \Delta_0} \quad \dfrac{\dfrac{\dfrac{}{\bullet : \emptyset}}{var\,y : int, \bullet : \emptyset, [y, int\ loc]}}{var\,x : bool, var\,y : int, \bullet : [y : int\ loc, x : bool\ loc] = \Delta_2}}{\Delta_0 \vdash D_1 : \Delta_1 = [\rho : bool, int, \bullet\ proc]}}{\emptyset \vdash D : \Delta : \Delta_0[\Delta_1] = [z : int\ loc][\rho : bool, int, \bullet\ proc]} \quad \dfrac{\dfrac{\Delta_0[\Delta_2] \vdash y : int \quad \Delta_0[\Delta_2] \vdash 2 : int}{\Delta_0[\Delta_2] \vdash x : bool} \quad \Delta_0[\Delta_2] \vdash C_2, \Delta_0[\Delta_2] \vdash C_3}{\Delta_0[\Delta_2] \vdash C_1}}{\emptyset \vdash C}$$

PROCEDURE - SEMANTICA DINAMICA 14/4/14

C ::= ... | p(ae)

D ::= ... | procedure p(form) C

form ::= • | const x : τ, form | var x : τ, form

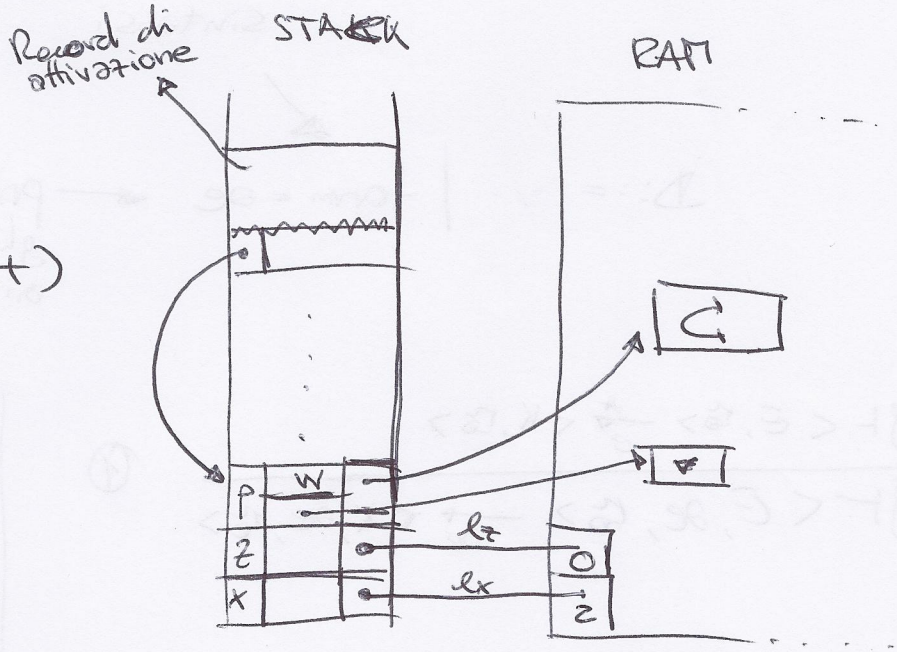
ae ::= • | E, ae

abs ::= λ.form.C

$\rho \vdash \langle \text{procedure } p(\text{form})\ C, \sigma \rangle \xrightarrow{d} \langle [\rho : \lambda\, \text{form}, C'], \sigma \rangle$

es

```
var x : int = 2;
var z : int = 0;
procedure p(var w:int)
        z := x + w
.
.
p(z)
```



NB: nel caso di scoping statico: ricordarmi nel record della dichiarazione di p la definizione delle variabili libere disponibili prima della dichiarazione della procedura => aggiungere un puntatore che punta alle variabili nello stack che usa il corpo della funzione. Nel caso di scoping dinamico non ne ho bisogno perché vado a controllare il valore delle variabili al momento della chiamata.

(*)

$$C' = \begin{cases} \rho_{|FI(C)\setminus form_i}\ C & \text{scoping statico} \\ C & \text{scoping dinamico} \end{cases}$$

↳ ro ristretto agli identificatori liberi di C

$\rho \vdash \langle p(ae), \sigma_0 \rangle \xrightarrow[c]{} \langle form = ae ; C, \sigma_0 \rangle$, $\rho(p) = \lambda form. C$

↑ non è dichiarato nella sintassi

$D ::= \ldots \mid form = ae$ ← prima devo valutare tutti gli ae e poi associarli ai form.

$$\frac{\rho \vdash \langle E, \sigma_0 \rangle \xrightarrow[e]{*} \langle K, \sigma_0 \rangle}{\rho \vdash \langle E, ae, \sigma_0 \rangle \xrightarrow[ae]{} \langle K, ae, \sigma_0 \rangle} \quad (1)$$

$$\frac{\rho \vdash \langle ae, \sigma_0 \rangle \xrightarrow[ae]{} \langle ae', \sigma_0 \rangle}{\rho \vdash \langle K, ae, \sigma_0 \rangle \xrightarrow[ae]{} \langle K, ae^{+1}, \sigma_0 \rangle} \quad (2)$$

$$\frac{\rho \vdash \langle ae, \sigma_0 \rangle \xrightarrow[ae]{*} \langle aK, \sigma_0 \rangle}{\rho \vdash \langle form = ae, \sigma_0 \rangle \xrightarrow[d]{} \langle form = aK, \sigma_0 \rangle} \quad (3)$$

$$\frac{aK, L \vdash form : \rho_0, \sigma_0}{\rho \vdash \langle form = aK, \sigma_0 \rangle \xrightarrow[d]{} \langle \rho_0, \sigma_0 \rangle} \quad (4)$$

(5)

$aK ::= \bullet \mid K, aK$

$\bullet, L \vdash \bullet : \emptyset, \emptyset$

$$\frac{aK, L \vdash form : \rho_0, \sigma_0}{(K, aK), L \vdash const\ x : \tau,\ form : \rho_0[x = K], \sigma_0}$$

$$\frac{aK, L \setminus \{l\} \vdash form : \rho_0, \sigma_0}{(K, aK), L \vdash var\ x : \tau,\ form : \rho_0[x = l],\ \sigma_0[l = K]}$$

es

const y : bool = tt ;] D_0
var x : int = 2 ;] D_2
var z : int = 0 ;] D_3] D_1] D] C
procedure p (var w : int) (form)
C_1 [if y then z := x + w (C_2)
else z := x * w (C_3)] D_1
p(z)] C_0

$\emptyset \vdash \langle C, \tilde{\sigma}_0 \rangle \xrightarrow{c}$

$\langle [y:tt] ; D_1 ; C_0 , \tilde{\sigma}_0 \rangle \rightarrow \langle [y:tt] : [x:l_x] ; D_1 ; C_0 , \tilde{\sigma}_0[l_x = 2] \rangle \rightarrow$

$\langle [y:tt][x:l_x] ; D_1 ; C_0 , \tilde{\sigma}_0[l_x=2] \rangle \rightarrow \langle [y:tt][x:l_x] ; [z:l_z] ; D_1 ; C_0 , \tilde{\sigma}_0[l_x=2][l_z=0] \rangle$

$\rightarrow \langle \underbrace{[y:tt][x:l_x][z:l_z]}_{\rho_0} ; D_1 ; C_0 , \underbrace{\tilde{\sigma}_0[l_x=2][l_z=0]}_{\tilde{\sigma}_{00}} \rangle$

sc din $\rightarrow$? $\langle \rho_0 [p : \lambda \text{ var } w : int . C_1] ; C_0 , \tilde{\sigma}_{00} \rangle$

$\langle \underbrace{\rho_0 [p : \lambda \text{ var } w : int . C_1]}_{\rho_1} ; \text{var } w \text{ int} = z ; C_1 , \tilde{\sigma}_{00} \rangle$

sc stat? $\rightarrow$ $\langle \rho_0 [p : \lambda \text{ var } w : int . \rho_0 ; C_1] ; C_0 , \tilde{\sigma}_{00} \rangle$

$\langle \underbrace{\rho_0 [p : \lambda \text{ var } w : int . \rho_0 , C_1]}_{\rho_1} ; \text{var } w : int = z ; \rho_0 ; C_1 , \tilde{\sigma}_{00} \rangle$

sc din $\rightarrow$ $\langle \rho_1 ; [w = l_w] ; C_1 , \tilde{\sigma}_0[l_w = 0] \rangle$

sc stat $\rightarrow$ $\langle \rho_1 [w = l_w] ; \rho_0 ; C_1 , \tilde{\sigma}_0[l_w = 0] \rangle$

sc div →

$$\langle \rho_1[w=l_w]; c_2, \tau_0[l_w=0] \rangle =$$

$$\langle \rho_1[w=l_w], \tau_0[l_w=0][l_z=z] \rangle$$

sc stat →

$$\langle \rho_1[w=l_w] f_0; c_2, \tau_0[l_w=0] \rangle$$

$$\langle \rho_1[w=l_w] f_0; \tau_0[l_w=0][l_z=z] \rangle$$

17/4/14

Es 1: Scrivere 2 programmi IMP, diversi, ma che siano equivalenti. Dimostrarlo.

Es 2:

$C[$ var z : int = 1 ; $]_{D_0}$
$($ var x : int = 5 and procedure p(var y : int$_{rw}$) z := x + y $)_{D_2}$ $_{D_3}$
p(2) ; $]_{C_1}$ $]_{D_1}]_{C_2}$

Determinare il valore di x e z dopo la chiamata di proc. in caso di ~~semantico~~ scoping statico e din. Motivare le risposte.

Es 3:

$D_1[$ var x : int = 3
$C[$ $(($ var z : int = x+1 $)_{D_5}$ and const y : int = 5 $)_{D_6}]_{D_3}$
in procedure p(var y : int)
var x : int = z ; z := x + z + y $]_{D_4}$
p(3) ; x := z+1 ;

Determinare il valore di x dopo l'ultimo := in caso di sc. stat. e din. Motivare la risposta evidenziando le derivazioni principali.

Es 2 scoping din, semantica din

$\emptyset \vdash \langle C_1, \emptyset \rangle \rightarrow \langle [z = l_z]; C_3, \emptyset[l_z = 1] \rangle$ (3) $(C_1 + C_4)$

$\langle [z = l_z][x = l_x][p = \lambda var\ y : int.\ C_2]; C_1, [l_z = 1][l_x = 4] \rangle \longrightarrow$

$\langle \underbrace{[z = l_z][x = l_x][p = \lambda var\ y : int.\ C_2]}_{\rho_0}; var\ y : int = 2;\ z := x + y,\ [l_z = 1][l_x = 4] \rangle \xrightarrow{2}$

- valuto x
- valuto y
- somma
- scrivo in l_z

$\langle \rho_0 [y : l_y]; z := x + y,\ [l_z = 1][l_x = 4][l_y = 2] \rangle \xrightarrow{4} \langle \rho_0 [y : l_y],\ [l_z = 6][l_x = 4][l_y = 2] \rangle$

Es 1

var x: int = 1
var y: int = 2
(x = x+1 and y = y+1)

---

var x: int = 2

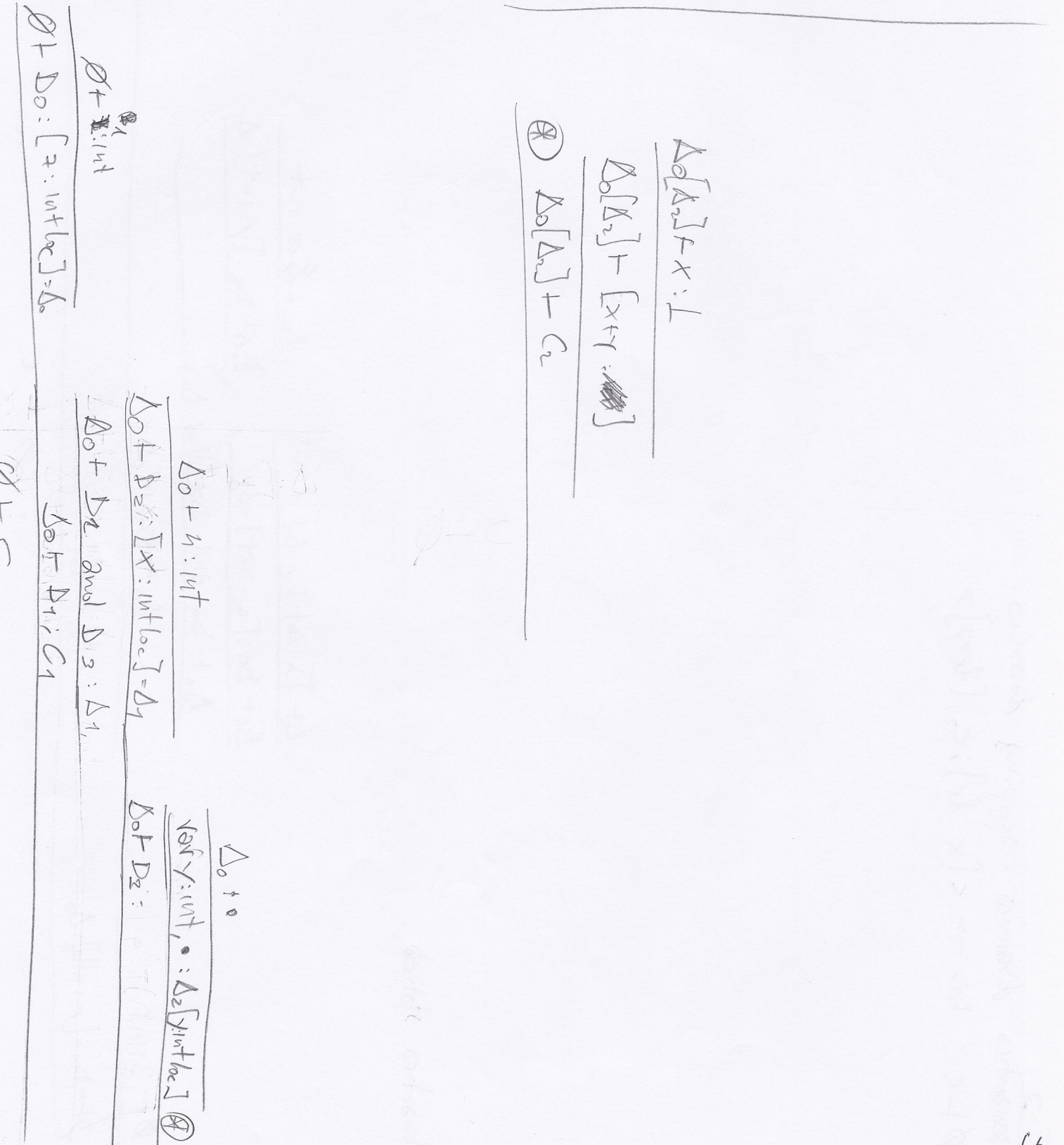
var x: int = 1
var y: int = 2
(y = y+1 and x = x+1)

---

var x: int = 2

C_0 = nil
C_1 = nil ; nil
$C_0 \equiv C_1 \iff \forall \sigma . Exec(C_0, \sigma) = Exec(C_1, \sigma)$
=> nil non modifica la memoria

Es 2



Es 3

semantica dinamica - scoping dinamico.

$\emptyset \vdash \langle C, \sigma_0 \rangle \rightarrow \langle [x : l_x]; C_2, [l_x = 3] \rangle$

Semantica statica

$$\frac{\dfrac{\Delta_0 \vdash [x : int] \quad \Delta_0 \vdash 1 : int}{\Delta_0 \vdash D_5 : [z : int] = \Delta_1} \qquad \dfrac{\Delta_0 + \sigma_0 : int}{\Delta_0 \vdash D_6 : [y : int] = \Delta_2}}{\dfrac{\Delta_0 \vdash D_5 \text{ and } D_6 : [\Delta_1, \Delta_2]}{\Delta_0 \vdash D_3 \text{ in } \Delta_1} \qquad \dfrac{\emptyset \vdash 3 : int}{\emptyset \vdash D_1 : [x : int] = \Delta_0}}$$

$$\frac{\Delta_0 \vdash D_2 \qquad \emptyset \vdash C}{\emptyset \vdash C}$$

11/04/2014

$C ::= \ldots \mid P(ae)$

$D ::= \ldots \mid$ procedure $p(form) \subset \mid form = ae$

$form ::= \bullet \mid$ var $x : \tau$, $form \mid$ const $x \tau$, $form$

$ae ::= \bullet \mid E, ae \qquad aet ::= \bullet \mid \tau, aet$

$abs ::= \lambda\, form . ae$

$ak ::= \bullet \mid K, ak$

PASSAGGIO DI PROCEDURE COME PARAMETRI

⇒ estensione di form con proc $p : aet_{proc}$, form ①

" " ak con abs, ak ②

" " aet con aet_{proc}, aet ③

$T(\text{proc } p : aet_{proc}, form) = aet_{proc}, T(form)$ ①

$FI_{form}(\text{proc } p : aet_{proc}, form) = \emptyset$

$$\frac{form : \Delta_0}{\text{proc } p : aet_{proc}, form : \Delta_0[p : aet_{proc}]}, \quad \Delta_0 : I_0, \; p \notin I_0$$

(7)

$$\frac{\partial K, L \vdash form : \mathcal{S}_0, \tilde{\tau}_0}{(\lambda form.C, \partial K), L \vdash proc\ p : setproc, form : \mathcal{S}_0[p : \lambda form.C], \tilde{\tau}_0}$$

(8)

$$\frac{\mathcal{S}.\mathcal{S} \vdash \langle \partial e, \tilde{\tau}_0 \rangle \xrightarrow{\partial \varepsilon} \langle \partial e', \tilde{\tau}_0' \rangle}{\mathcal{S} \vdash \langle \lambda form.C, \partial e, \tilde{\tau}_0 \rangle \longrightarrow \langle \lambda form.C, \partial e', \tilde{\tau}_0' \rangle}$$

PASSAGGIO PARAMETRI PER RIFERIMENTO.

$form ::= \ldots \mid ref_\tau\ x:\tau, form$ $\quad$ $det ::= \ldots \mid \tau_{loc}, det$

$dK ::= \ldots \mid l : dK$ $\quad$ (modi) $\mu ::= \bullet \mid val, \mu \mid ref, \mu$

$T(ref\ x:\tau, form) = \tau_{loc}, T(form)$

$FI_{form}(ref\ x:\tau, form) = \emptyset$

$\Pi(\bullet) = \bullet$

$\Pi(\tau, det) = val, \Pi(det)$

$\Pi(\tau_{loc}, det) = ref, \Pi(det)$

$$\frac{form : \Delta_0}{ref\ x:\tau, form : \Delta_0[x:\tau_{loc}]}$$

$I_0 : \Delta_0,\ x \notin I_0$

$\Delta_0(x) = 1$

$$\rho \vdash_{ref,\mu} \langle (x, de), \sigma_0 \rangle \xrightarrow{de} \langle (l, de), \sigma_0 \rangle,\ \rho(x) = l$$

$$\rho \vdash_{val,\mu} \langle (x\ de), \sigma_0 \rangle \xrightarrow{de} \langle (\kappa, de), \sigma_0 \rangle,\ \rho(x) = l,\ \sigma_0(l) = \kappa$$

$$\frac{\rho \vdash_\mu \langle de, \sigma_0 \rangle \xrightarrow{de} \langle de', \sigma_0' \rangle}{\rho \vdash_{ref,\mu} \langle (l, de), \sigma_0 \rangle \xrightarrow{de} \langle (l, de'), \sigma_0' \rangle}$$

$$\frac{dK, L \vdash form : \rho_0, \sigma_0}{(l, dK), L \vdash ref\ x:\tau, form : \rho_0[x:l], \sigma_0}$$

$$\frac{\mathcal{G} \vdash_{\mu} \langle \partial e, \hat{\sigma} \rangle \xrightarrow{\partial \alpha}{}^{*} \langle \partial K, \hat{\sigma}' \rangle}{\mathcal{G} \vdash \langle form = \partial e, \hat{\sigma} \rangle \xrightarrow{\alpha} \langle form = \partial e, \hat{\sigma}' \rangle} \quad \mu = \Pi(T(form))$$

Quale è la differenza tra i linguaggi imperativi e funzionali?

~~[illegible]~~

Applicazione di funzione su valori

Mancanza di Memoria solo ambienti.